HEWLETT PACKARD

RTE-A
System Manager's Manual

Data Systems Division
1266 Kifer Road
Sunnyvale, CA 94086-5304

Manual Part No. 92077-90056
E0887

Printed in U.S.A. August, 1987
First Edition

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THE MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Printing History

The Printing History below identifies the edition of this manual and any updates that are included. Periodically, update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this printing history page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past updates; however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all updates.

To determine what manual edition and update is compatible with your current software revision code, refer to the Manual Numbering File or the Computer User's Documentation Index. (The Manual Numbering File is included with your software. It consists of an "M" followed by a five digit product number.)

First Edition ..... Aug,1987 ..... Rev. 5000 (Software Update 5.0)

Preface

The RTE-A System Manager's Manual describes the duties of a system manager. It is designed to complement the RTE-A System Generation and Installation Manual.

The manual gives a brief overview of system management and refers to the other manuals in the RTE-A manual set for detailed information on specific elements of the system. Particularly useful are the System Design Manual, the Utilities Manual, the Driver Reference Manual, and Programmer's Reference Manual for maintenance and expansion.

The manual also describes resource management in the multiuser environment through the Group and User Management Program (GRUMP) utility, the Security/1000, and the SECTL utility. Note that information on system security tables and the Security/1000 library (SECLIB.1000) are contained in this manual so that the system manager can control the distribution of this information.

Conventions Used in this Manual

The following conventions are used in this manual:

Convention	Meaning
Uppercase letters	In command syntax, uppercase letters indicate characters that must be entered as shown.
Lowercase letters	In command syntax, lowercase letters indicate user-supplied information.
[ ]	In command syntax, brackets indicate optional items.
	In examples of interactive sessions, brackets indicate default values.
Underscore	In examples of interactive sessions, all user input is underscored.
, or blank	A comma or blank are command delimiters.

Table of Contents

Chapter 1
System Management Overview

Chapter 2
File System Security

Chapter 3
Multiuser Account System

Appendix A
SECURITY/1000 Error Codes

Appendix B
SECURITY.TBL

Appendix C
SEC1000.LIB

Appendix D
D Sample Answer File

Appendix E
E Logon Files

Appendix F
GRUMP Command/Log Files

List of Illustrations

Tables

1

System Management Overview

The 92077A RTE-A Operating System Software is a powerful and flexible operating system providing an environment that allows concurrent user access to system resources. As system manager, your duties include:

- Design and Planning - determining the system requirements and structure; deciding whether or not to use the primary system; installing the primary system if it is to be used.
- Generation and Installation - creating a customized system from the primary system if there is no existing system; getting the new system running.
- Maintenance - maintaining the existing system's operation and integrity by regenerating as required; performing backup of the operating system; performing backup of user files and applications; keeping the system current with software updates; answering questions about system operation; keeping software and hardware documentation current and available.
- System Recovery and Shutdown - recovering system operation from the backup system copies made at installation if needed; performing system shutdown as needed.

The process of system management is described by the flowchart in Figure 1-1.

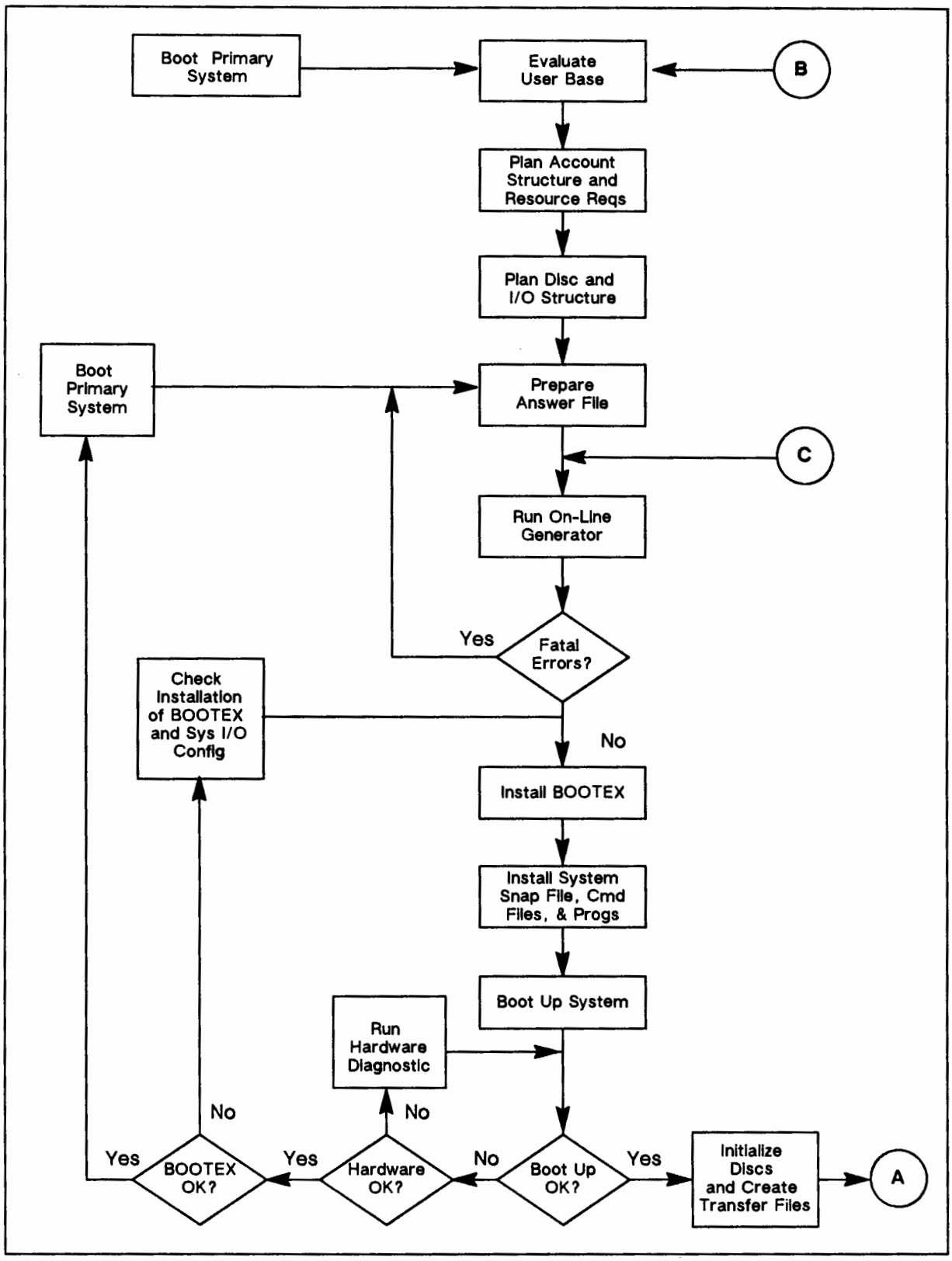

Figure 1-1. System Management Procedural Overview

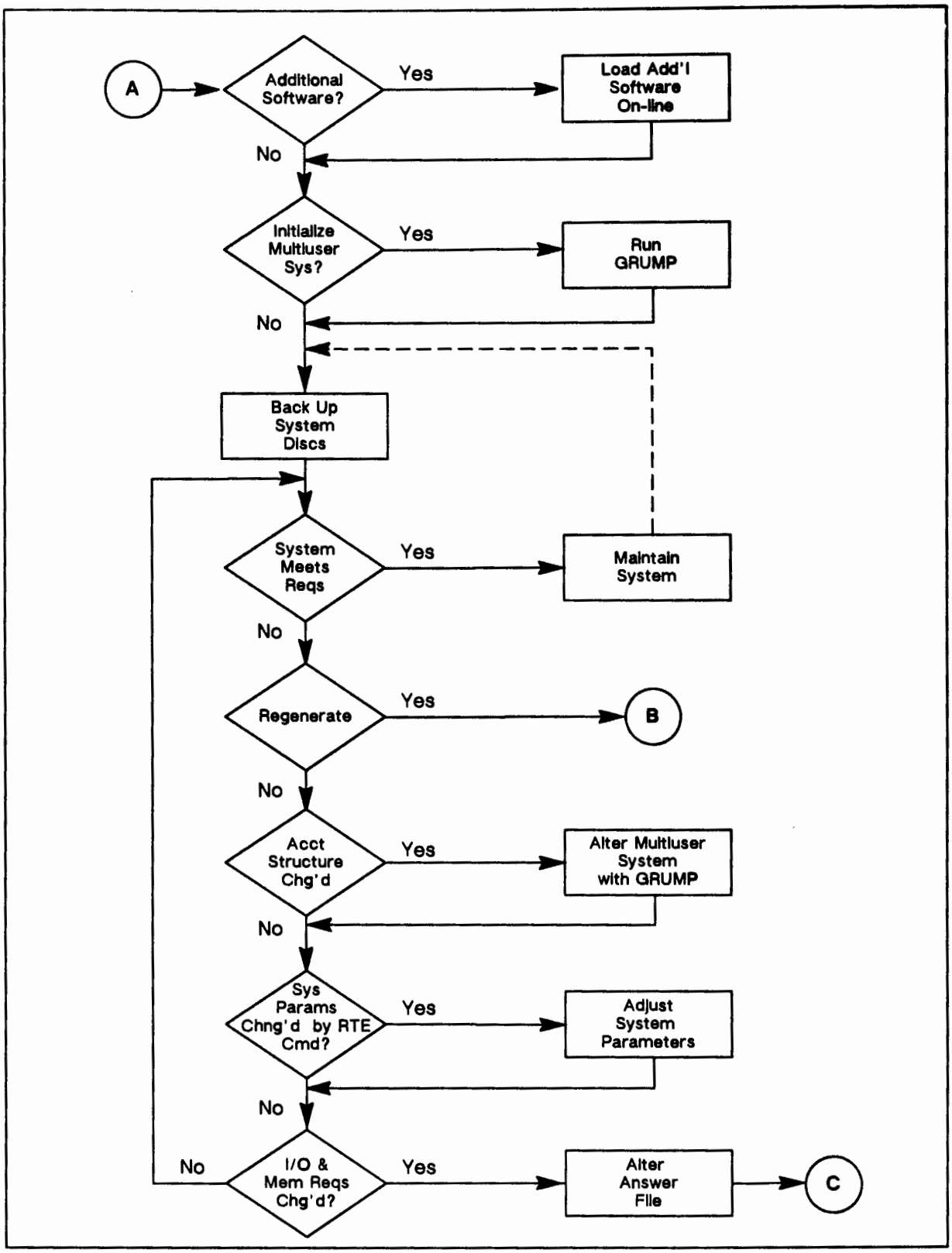

Figure 1-1. System Management Procedural Overview (cont'd)

Design and Planning

In planning system requirements, system managers need to know:

- Who will be using the system
- What applications will be run
- What system resources and peripherals will be required

You must consider types of applications to be run, required system resources and peripherals, and possible future expansion. Refer to the RTE-A System Design Manual (Part No. 92077-90013) for detailed information on system design concepts, software planning and I/O design and planning.

The steps in system planning are:

- Plan the account structure and the disc volume and cartridge requirements.
- Plan the computer I/O structure, determining select codes for each card, LU numbers for each device, and the drivers required to control these cards and devices. See the RTE-A System Generation and Installation Manual (Part No. 92077-90034) for details.
- Plan the RTE-A memory allocation by determining the number of reserved partitions, size of SAM, size of XSAM, number of ID segments, class numbers, and resource numbers.

Determining User Requirements

To determine user requirements, you should create a questionnaire and use it to interview potential system users. The questionnaire in Figure 1-2 can be used as a guide but should be modified to your specific needs. Most users do not think in terms of disc tracks, memory or disc-resident programs, or priority levels when describing their needs. Therefore, the questions should be readily understood and provide the kind of information that can be translated into data useful for system generation, initialization, and maintenance.

The topics any questionnaire should cover include the following:

- User Category
- Applications
- Peripheral Resource Usage

USER CATEGORY

_____ General Users
_____ Junior Operators
_____ Senior Operators
_____ Application Programmers
_____ System Programmer/System Manager

} Suggested User Categories

Set Categories Appropriate to Your System

# of concurrent users _____

User Names _____ _____ _____

APPLICATIONS

Special Program Requirements

EMA Programs

Size ___ Should EMA be shareable? Y N With what programs? _____

# of class Numbers _____

# of words of SAM and XSAM _____

# of Resource Numbers _____

# of shared programs _____

Number of programs active at one time _____

Labeled Common Size _____

Unlabeled Common Size _____

Program partition size _____ Partition reserved? Y N

PERIPHERAL RESOURCE USAGE

Subsystems _____

_____ Using hierarchical file system
_____ Using FMGR file system
_____ Common Data Base/File Access
_____ Line Printer Access
_____ Cartridge Tape Drive Access
_____ Magnetic Tape Unit Access
_____ Others
_____ Special Requirements

} Suggested Peripherals

Set Categories Appropriate to Your System

Figure 1-2. Determining User Requirements

User Categories

The first section helps define levels of user sophistication. This section is primarily applicable to systems with VC+. Five levels of user sophistication are defined in this sample, but you should customize these categories to your system.

General users are at the first level of user sophistication. Users in this group interface with the system only to the extent of operating specific programs or command files. No programming knowledge is necessary and very little knowledge of the system is required. Users are expected to follow predefined procedures when working with the computer.

Junior and senior operators are at the next two levels. Users at these levels may require knowledge of the editor and cursory knowledge of the file system. Only limited access to the system functions is needed.

The application programmer is at the fourth level. Most RTE-A users will fall into this category. These users have knowledge of operator commands, programming calls, etc. They are expected to take advantage of most system capabilities including operation of compilers, management of data bases, manipulation of the file system, performance of network operations, etc. However, these users are not concerned with the activities of the other users on the system. Detailed system knowledge is not usually required.

Users at the highest level of sophistication include system programmers and support personnel. These users have a good working knowledge of system operation. They are capable of changing overall system operating parameters.

The user categories information can be the basis for groups. You should also determine the number of users who will be on the system at the same time (concurrent users). Refer to Chapter 3 for information on the multiuser account system.

System Applications

This section deals with intended system applications. The answers to questions in this section decide how system resources will be allocated and how to set up system parameters.

The Applications and Peripheral Resource Usage sections apply to each user individually. All information such as number of class numbers or program partition size should be collected for each individual application and tabulated for the total system requirement. If system size allows, extra resources should be added to the total user requirement for use by future applications.

Answers collected here will determine:

- Subsystems required. The HP-supplied subsystems, languages, utilities, and user application programs to be used on the system. Determine which subsystems (for example, DS or IMAGE) will be used because all of them have individual system requirements. Encourage users to consider future expansion as well as current needs.

- Response time requirements. Users should be asked their terminal and real-time response requirements. Based on their responses, modules may be given higher priority levels and/or assigned to partitions. For example, response considerations in a real-time environment may dictate that certain programs be memory-resident at all times. These programs may need to have priorities higher than the priority of the Real Time Fence. Also, you will use the information about these programs to determine sizes of reserved partitions.
- Memory Requirements. You must have information about real-time programs when creating a boot command file so that you can create reserved partitions large enough to accommodate those programs. Partitions must be large enough to accommodate any large application programs users will run. Use the number of reserved partitions and user programs to adjust the number of the memory descriptors specified in the generation file. Extended Memory Area (EMA) usage is another factor to be considered. User applications using the EMA feature require partitions of a certain minimum size. This affects the amount of physical memory required in the system. Therefore, users should be asked for the maximum EMA space used in application programs and whether or not the information in the EMA is to be shared by more than one program.

Peripheral Resource Usage

This section concerns required peripheral resources. The peripherals shown in Figure 1-2 are suggestions only. You should tailor the list of peripherals to your own system. Each user or group of users must provide the following information:

- Will the user be storing files or creating data bases on the system? If so, how many and how big? Does the user require disc space on a permanent or temporary basis? This indicates the amount of disc space (if any) to allocate the user.
- Will the user's files be accessed by other users in the system? Will this user access other users' files? Which users? Does this user have files that cannot be shared? These questions are important in systems using VC+ because file access can be restricted to the individual users, made available to all system users, or made available to members of a group.
- Does the group require a special peripheral? For example, a peripheral may be necessary for one group's application. Another group on the same system but involved in a different application may wish to discourage the use of that same peripheral.
- Will the users need the screen editor (EDIT) for program development or other text processing? If so, adequate disc space should be made available to accommodate the scratch files created by EDIT. You can dedicate a disc volume to storing temporary editor scratch files. Refer to the EDIT/1000 User's Manual for editor loading information. You must also make disc space available for other programs such as LINK and TF which require large amounts of space for scratch files.
- Does the disc have an integrated Cartridge Tape Drive (CTD)? If so, reserve a buffer area on the disc for the CTD. This area is referred to as the disc cache.

The size of the disc cache for the CTD operations greatly improves the data transfer rate. Refer to the RTE-A System Generation and Installation Manual (Part No. 92077-90034) for information on how to reserve this area.

After determining the devices that will be used by your applications, assign them LUs. Integrated devices have special requirements because each device needs a separate LU number (and in some cases, a separate driver). Some applications require LUs that are less than 63. (There is a maximum of 255 LUs.) Conventions in assigning LUs to logical devices are:

- System console = LU 1
- Line printer = LU 6
- Streaming magnetic tape = LU 7
- Magnetic tape = LU 8
- CS/80 cartridge tape = LU 24
- Disc LUs = LUs up to and including 63
- Maximum LU number = 255

Find out whether or not there are any special needs such as time requirements or access restrictions. For example, one group may need to use the magnetic tape drive for several hours each night; another group needs the same tape drive early in the morning, and still another group needs that tape drive available most of the day. Coordinating user needs is very important when determining peripheral requirements.

Disc Management Considerations

In RTE-A, discs and files are managed by two file system interfaces, CI (Command Interpreter) and FMGR. In CI, the preferred interface, files and directories are managed and protected by read/write protection. The CI, or hierarchical, file system divides the disc into large areas of free blocks. These areas are identified by LU numbers and are called disc volumes. Files in each disc volume are managed by directories and subdirectories which maintain information on the files.

The FMGR file system divides a disc into fixed-size cartridges that are identified with cartridge reference numbers (CRN). The CRN can also be a two-character string. Each cartridge has a cartridge directory containing pertinent information on all files stored on that cartridge. Although CI is the preferred file management system, it is useful to have at least one FMGR cartridge in case some utility requires one.

Table 1-1 shows a comparison of the two file systems. The CI file system lets you assign several users to one large disc volume, making the free space on the volume available to all users. Usually, each user is given a private set of hierarchical directories for file management. As shown in Table 1-1, CI provides time stamps for file creation time, last update and last access; it provides file unpurge capabilities, and file names up to 16 characters long. FMGR, on the other hand, provides only one directory per disc LU, and the files on each LU must have unique names limited to six characters.

Table 1-1. RTE-A File System Comparison

	CI File System	FMGR File System
File name	1–16 characters	1–6 characters
Cartridge/ directory	1–16 characters in directory	1–2 characters or numeric cartridge names
File Security	Protection based on directory and file ownership	Security code used for file protection
File Types	File type extensions describing the contents of the files	Defines the structure of the files
File Mask	Mask qualifier and special characters in file name	Limited file masking
File Size	Extendible	Extendible
Time Stamps	Create, access, and update times handled by the file system	None
Subdirectory	Subdirectories within directories and other directories	None
File recovery	Operator recoverable immediately after purge	None
Spooling	Can be done interactively and programmatically	Can be done interactively and programmatically
Incremental Backup	Done in conjunction with TF and FST utilities	None

File Volume

A volume is a self-contained section of a disc. Each volume is independent of any other volume. Directories and files can never cross volumes. Each physical disc drive consists of one or more volumes. Volumes can never cross physical drives. Each volume is identified by a logical unit (LU) number. Volumes are always identified by their disc LU number. Volume LU numbers range from 1 to 63 inclusive.

Each volume contains information about the layout. This information includes the names of all the global directories in the volume, as well as a table that tells which disc blocks have been allocated in the volume. This table is called a bit map because the table is composed of bits rather than addresses or values.

When deciding on a disc layout, be aware of the advantages and disadvantages of using a single LU or several large LUs. If you use a single LU, you will benefit from initial low maintenance and some applications such as a data base file require a large unit. However, the single LU tends to waste disc space. Using several large LUs has the advantages of using space better with less waste, being easier to pack, and being easier to work with. In general, it is better to have a small number of large disc LUs. Make sure that your scratch file and swap file are on large LUs for system operation purposes.

Some advantages of larger LUs are:

- There is more room for files and subdirectories under one global directory
- The disc does not need packing as often
- There is space for very large files is easier to find (for scratch and swap files, for example)

Some advantages of smaller LUs are:

- The backup of an individual LU is faster
- Required FMGR LUs can be allocated a smaller amount of disc space
- Users can be restricted to specific LUs. The advantage here is that you can use security to restrict users to LUs and volume ownership can be specified.

Mounting a volume makes that volume and all the directories and files on it available to the operating system. Dismounting a volume removes that volume and makes the directories and files on it inaccessible to the system. These operations are not performed frequently except with removable media such as floppy discs where discs must be mounted after they are installed and dismounted before they are removed.

Mounting a volume will initialize it if there is no valid data on the volume. Initializing a volume sets up information needed by the operating system including the list of directories and the bit map for keeping track of space use. If the disc volume does not have a valid FMP or FMGR directory, you are prompted before the volume is initialized. This prevents accidentally corrupting volumes that are not FMP or FMGR types (for example, backup utility volumes). If the disc volume does have a valid FMP or FMGR directory, the volume is initialized.

Except for duplicate directory names on two or more volumes, the order in which disc volumes are mounted is unimportant. If a global directory on the newly-mounted disc volume has the same name as a previously mounted global directory, the new directory is inaccessible.

The CI MC command does not place reserved blocks at the beginning of the volume. If reserved blocks are required, use the CI IN command. For more information on CI file volumes, refer to the RTE-A User's Manual (part no. 92077-90002).

Directory Organization

Directories are the central CI file system data structure. Directories maintain the file system state across system boots. All information pertaining to a file is maintained in a directory.

Directories may be included in other directories, these are considered subdirectories. Nesting of subdirectories is allowed to provide a hierarchical file system structure. At the top is a root directory that contains only unique global directories. There is one root directory per disc volume. Mounting a disc volume makes directories on that volume accessible.

Global directory names must be unique in the file system. An abbreviated form of the directory name is kept in free space in D.RTR; each global directory requires five words. For this reason, a limit of approximately 300 global directories is recommended. This limit applies only to global directories and not to subdirectories. Therefore, convert global directories into subdirectories (with the CI MO command) if the 300 limit is reached.

Generation and Installation

If your system is already running, refer to the Maintenance section.

If you are installing a system for the first time, the RTE-A Primary Installation Manual (part no. 92077-90038), which comes with the software, provides the procedure for installing a primary system.

The primary system is a tested, factory-configured operating system that can be booted up immediately after installation. It provides a working starter system that can be used to test the functions of the installed hardware. It can be used by the system manager to regenerate a customized system. It may also be used by all levels of users to gain familiarity with the RTE-A operating system features.

Each primary system tape includes all the RTE-A operating system components needed to generate a new system. The Software Numbering File (A92077) is a text file which lists all the files included on the primary tape. Note that the primary system tape does not include the VC+ software. You must regenerate if you use VC+.

Once your primary system is running, the steps in installing your disc-based system are:

1. Preparing the boot LU by creating a BOOTEX area, if necessary.

   BOOTEX is a memory-based system that has the sole responsibility for loading loading and initializing a disc-based operating system.

2. Installing BOOTEX, if necessary.

3. Preparing the boot command file.

   The boot command file contains commands for the BOOTEX system to initialize the disc-based system.

4. Installing system, snap, and boot command files on boot LU.

5. Creating the required directories and program files.
6. Setting up startup program and the WELCOME file.

   The WELCOME file is a CI command file that is run by the RTE-A operating system to mount volumes, initialize devices, and so forth.
7. Booting and initializing the system.
8. Verifying operation and backing up the system.

Generation

System generation consists of preparing a system generation answer file and running RTAGN, the RTE-A online generator. An answer file contains prepared responses to generator program questions when building the new system. Normally, an existing answer file (a sample answer file is shipped with the RTE–A operating system) is edited to create a new answer file. See the RTE-A System Generation and Installation Manual for a sample of an answer file. Appendix D in this manual shows a portion of a sample answer file used in generating a system with Security/1000.

The RTAGN program produces the system, snapshot and list files required to define the system to be installed. The system file contains a memory image of the operating system. At system bootup, the system file is copied from disc or other bootable media to physical memory and executed.

The snap (snapshot) file contains the value of system entry points, system library names, and other system information such as system checksums and system common checksum. This is used by LINK to load programs on-line. The current snap file must be copied to the system directory and named SNAP.SNP because LINK automatically searches for that name.

The list file documents what is in the system and where the modules are located. It contains:

- The runstring used to schedule RTAGN
- The system time at the beginning of the generation
- A listing of the input commands and comments
- A list of what was generated into the system and where the different parts of the operating system will be located
- The location and explanation of any errors that may occur

You should record and store new generation files and any changed generation files.

Refer to the RTE-A System Generation and Installation Manual for instructions on system generation.

Installation

System installation consists of generating a new system file, preparing the target system hardware and media for boot, booting the new system, setting up a primary program, establishing the account structure, spooling and directories (if VC+ is used), and backing up the new system.

Specific procedures are given in the RTE-A System Generation and Installation Manual.

Maintenance

You are responsible for maintaining the integrity of the running system and ensuring that it runs at optimal performance.

You, the system manager, must be prepared to:

- Alter the multiuser account system as required.
- Add features for new applications (for example, security and reserved memory).
- Backup and restore disc LUs and files.
- Alter system parameters to meet new user requirements or change generation parameters.
- Add on-line software (for example, subsystems such as graphics).
- Keep system and device documents current and available.
- Answer user questions about system operation.
- Interface with HP in problem reporting, resolution, and system updates (note that this function depends on the level of support a customer has purchased from HP).

If the changes warrant, the system may require a new generation. You will need to regenerate an existing system to add devices, modify tag size, change the number of ID segments or class numbers, add a subsystem, add or delete relocatables, or update the system with software revisions.

However, much maintenance (for example, altering time outs) can be done interactively. Ways to make a system more efficient or update it without regenerating include using the maintenance utilities on discs, files, and the accounting system, and performing the tasks listed in the fine tuning section below.

Accounting System

A system with multiple users and sessions is referred to as as multiuser account system. The GRoup and User Management Program (GRUMP) is used to create and maintain the multiuser account system. The GRUMP utility is presented in the Multiuser Account System chapter which also describes how to plan a system using groups. Use a questionnaire, such as the one in Figure 1-2, as a method of gathering the information needed for multiuser account planning.

Fine Tuning

To get optimal performance, you should:

- Minimize table space and base page links.
- Check device priority and location, making sure that important I/O devices are given higher priority than less important ones and that fast and slow devices are not connected to the same interface card. For example: a slow printer and a high speed disc should not be placed on the same HP-IB card.
- Determine the number of ID segments and class numbers that will control the number of system users without hindering those users' activities on the system.

Utilities for maintaining discs and files and keeping the system usable are described in detail in the RTE-A Utilities Manual. The disc utilities FORMC, FORMF, and FORMT are used to verify the integrity of discs, spare out bad areas of discs, and format and/or initialize discs. The file utilities FPACK, FREES, FOWN, FVERI, and FSCON report on the status of disc volumes and are used to manage usable space within the volumes. The FMGR PK command is used to pack FMGR cartridges, reclaiming space previously allocated to files that have been purged.

System Usability

You can use command files to set up or change special environments which make the system easier to use and enhance users' productivity. Global and user logon files, for example, are ways of enhancing system usability. A global logon file can be set up on /USERS to list a logon message or do other initialization, and then transfer to the user logon file. User logon files can be set up on the user's working directory to set up UDSPs, run a mail and/or phone program, and set up commonly used CI variables. You can initiate the logon file when specifying the user's startup program in the GRoup and User Management Program (GRUMP). An example is:

```
Enter the startup command [xxxx]: ru ci.run::programs /users/logon.cmd
```

where xxxx is the default value. Appendix E contains sample logon files.

System Backup

Backups are a method of saving data on a media other than the current disc (LU). Backup media includes CS/80 cartridge tapes, magnetic tapes, floppy discs, or another disc.

Doing system backup at initialization and periodically thereafter protects you and users from losing work and time in the event of unforeseen circumstances such as system shutdown. If system discs are corrupted or destroyed for example, you can recover by restoring a backup copy of the system.

Backing up your system can be done in two ways:

1 Physical backup that saves a physical snapshot of the disc and its exact contents.

2. Logical backup that looks for directory and file information on the disc and saves it as directories and files.

The comparison in Table 1-2 on the following page describes the differences between the two types of backup.

Table 1-2. Two Types of Backup

	Physical Backup	Logical Backup
Purpose	Save and restore disc LU on entire disc unit; save exact data on disc – does not have to be files (example, BOOTEX on a CI volume)	Save and restore files or groups of files only
Online/Offline	Online or Offline (utility dependent)	On-line only
Backup that can be used to boot system	Yes; usual format for primary systems	Not applicable
Utilities/Commands	ASAVE/ARSTR, DSAVE/DRSTR, Pushbutton save/restore IPBV, COPYL	FST, TF, FC, LIF, and CO commands in CI; ST and DU commands in FMGR
Advantages	Faster method if LU is full	Faster method if there are only a few files to be saved
	Restore entire LU	Restore on a file and directory basis

Primary/Physical System Backup

A physical backup is a copy of your system on a bootable medium or a medium that can be downloaded to the system disc with an off-line utility. You must have some form of physical backup or copy of your boot disc LU in case errors on the LUs with programs or the swap file result in system failure. This backup copy can be used to bring up a system quickly.

A physical backup on a disc is a bootable system. A physical backup on a magnetic tape or a CTD has to be downloaded to the disc before it can be booted. This can be done in one of the following ways:

1. Download with the pushbutton restore feature of CS/80 discs if you have a push-button save.
2. Boot a memory-based system which contains the corresponding physical restore utility. Then restore the necessary LUs from tape to disc.

You must prepare a bootable memory-based system which contains ARSTR (or your backup restore utility). This system must be available on some medium other than your system disc. This memory-based system is part of the physical backup.

The physical backup must be able to restore enough of the system to boot up, run the startup program, and bring other programs and files onto the system.

Physical backups can also be used to save other disc LUs besides the boot LU. They can be used to restore bad LUs in situations where your system is running and you can run ARSTR (or the restore utility corresponding to your backup utility) on-line. In these situations, the backup must be relatively current. You can use incremental logical backups to restore files that changed between the physical backup and the time that errors occurred on the LUs.

Note that physical backup utilities may not be compatible with different revisions of corresponding restore utilities. For example, a restore utility should not be relied on to work with a backup made with the corresponding save utility of a different revision. Also, do not rely on different physical backup utilities to be compatible with each other.

Physical backups should contain:

- Bootable BOOTEX
- System file
- Snap file
- Boot command file
- Welcome file
- System libraries
- Swap file
- Necessary directories such as /PROGRAMS and /SYSTEM
- Necessary programs such as CI, CIX, FMGR, D.RTR, D.ERR, EDIT, DL, LI, IO, WH,TF, FST, FSTP, FC, LINK, PROMT (for VC+), LOGON (for VC+)
- Message catalogs such as >TF000, >LK000, and >FS000
- Restore utility corresponding to your physical backup utility

Physical backups are not concerned with file structure. They save the physical image of data on the disc. The source and destination LUs must have the same physical characteristics. They must have the same track size and blocks per track. However, the total number of tracks in a disc LU need not match.

Logical Backup

A logical backup saves data on the disc on a per file basis. File structure and attributes are saved.

The FST utility is used to back up and restore files for both CI volumes and FMGR cartridges (except type 0 files) on magnetic tape or CS/80 cartridges. The TF and FC utilities are also used for backup and recovery but are not as fast as FST. Like FST, TF backs up both CI and FMGR files. FC is used only to back up FMGR files.

You can do a full backup which means a periodic backup of all files. You do an incremental backup by appending delta backups (backups of all files changed since the last backup) to the same tape as the full backup. The next full backup starts on a new tape. The advantages of doing incremental backups are: higher system availability since average delta backup time is faster; less tape used on the average; fewer tapes used since backups can be appended to the same tape; multiple versions of files can be accessed more conveniently for archiving. The disadvantages of incremental backups are that the files take longer to restore and the procedures require more effort to understand. Incremental backups are not applicable to FMGR files.

The FC utility is used to backup/restore FMGR files to magnetic tape or CS/80 cartridge tape. FC also can copy files between FMGR disc cartridges and tape media, either disc-to-disc, disc-to-tape, or tape-to-disc. FC, like TF, can be given a single command in the runstring, executed interactively, or given the name of a command file with a list of FC commands. TF cannot handle certain FMGR names (those which are not legal CI names) so it is a good idea to back up FMGR files with FC.

CI and FMGR commands are for disc-to-disc backup. Commands used in backups include the CI CO (copy) command and the FMGR ST (store) and DU (dump) commands. Although these CI and FMGR commands are available for disc-to-tape and tape-to-disc, use of FST, TF, or FC is more common.

The LIF (Logical Interchange Format) utility provides file interchange between an HP 1000 system and other HP computer systems.

System Backup Strategy

The steps in system backup are:

1. Building a memory-based version of your own primary system.

   After your system is running, build a primary system specifically tailored to your particular requirements (based on responses to a questionnaire similar to the one in Figure 1-2). Build a memory-based version of your system being sure to include D.RTR and ARSTR (or the restore utility corresponding to your physical backup utility). See the RTE-A System Generation and Installation Manual for details in building a primary system and creating a memory-based version of the system. Copy the system to tape (see the RTE-A Utilities Manual for details on copying to tape).

2. Doing physical backup of the system and other important LUs.

   Back up all files and programs necessary for minimal operation. Include programs necessary to bring other programs and files from your physical and logical backups.

3. Doing logical backup.

   Do periodic file backups using the FST or TF utilities.

The RTE-A Utilities Manual describes the physical backup and restore utilities and the logical backup utilities.

Keys To Successful System Backup

The following are recommendations for implementing a successful backup scheme:

- Maintain a strict backup schedule
- Keep users from accessing their files during system backup
- Only one person should clear the backup bit for a given set of files
- Label the tapes with date, time, contents, and type of backup
- Store backups in a safe place
- Keep system time accurate to maintain correct time stamps
- Use transfer files for backup and restoration to avoid errors
- Always verify backups and restores

System Recovery and Shutdown

Common situations in disc-based systems which require system recovery are system crashes, disc failure, or unsuccessful system installation. If you still have a bootable system in any of these situations, you can simply reboot. However, if you do not have a bootable system, you must boot from the primary system stored on tape. To recover a system backed up with ASAVE, use the ARSTR utility.

Once the system is running, use the FST, TF, and FC utilities to restore files and directories backed up with those utilities. With FST and TF, directories are restored to their original LU when possible. Directories restored by the superuser are owned by their original owner. Directories restored by a non-superuser are owned by the user doing the restore.

Situations may arise which require that you shut down the system. For example, you might shut down the system for hardware maintenance or new card installation. Adverse weather conditions require system shutdown because discs are susceptible to the power surges which may result from electrical storms. If you need to shut down the system, use the procedure recommended in the system installation and service manual.

You can turn the CPU power off when necessary without destroying data on the hard disc. Main memory will be lost unless your system is equipped with battery backup.

If you must power down your hard disc, power it down separately after CPU power is off following the instructions accompanying the disc. Refer to the operator manuals for all devices and follow the individual shutdown instructions.

2

File System Security

Security in the RTE-A system consists of two separate but overlapping subsystems, File System Security and Security/1000 (see Figure 2-1).

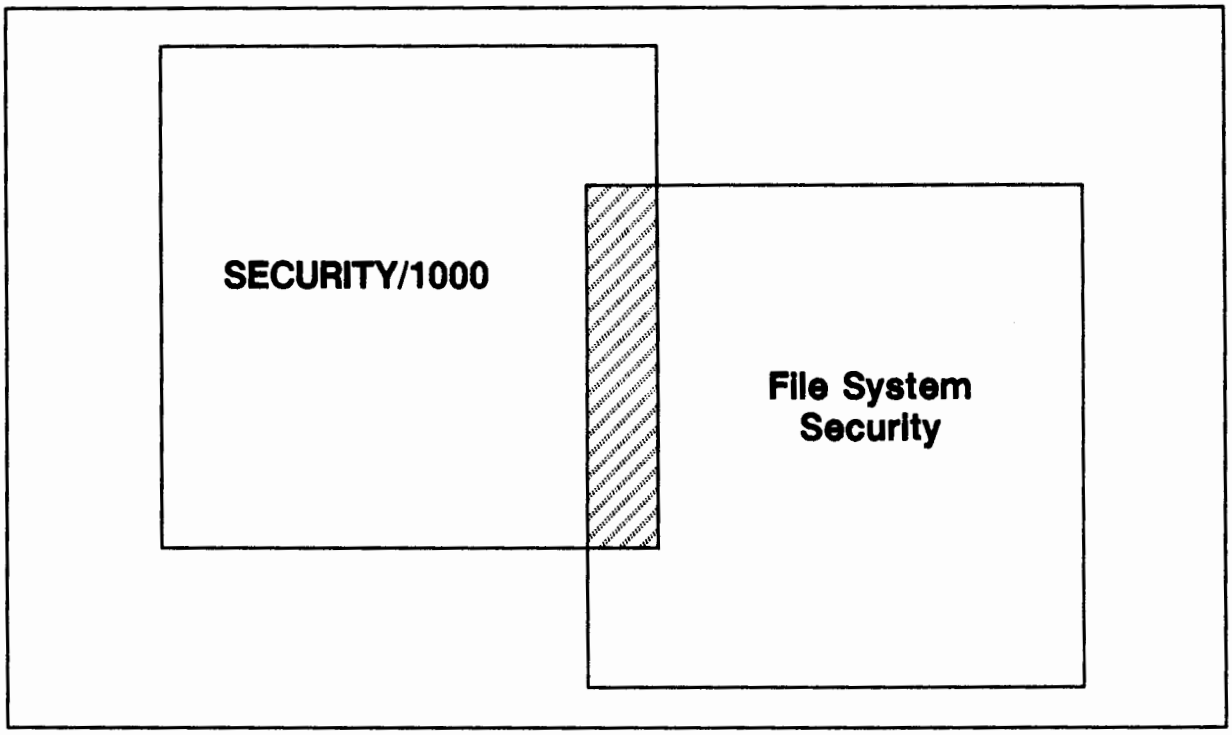

Figure 2-1. RTE-A Resource Protection

Both subsystems are built into the multiuser environment, but Security/1000 need not be turned on if the system manager does not want this checking mechanism. File System Security operates whether or not Security/1000 is turned on.

As system manager, you must decide whether or not Security/1000 is to be turned on, and if so, which Security/1000 configuration will be most applicable to your system.

The Security/1000 subsystem is described in detail at the end of this chapter.

File System Security

File System Security is defined as file, directory, and volume ownership and protection. In File System Security, users are divided into superusers and non-superusers. Superusers have a user capability level of 31 and are not subject to file system protection checks. Non-superusers have user capabilities from zero to 30, inclusive, and are subject to file system protection checks. As the system manager, you are a superuser and usually the only superuser. It is your responsibility to assign user capability levels.

As system manager, you have full system access including read/write access to any file (programs as well as text are files), directory, and volume on the system. You are the original owner of all system directories but can re-assign this ownership and associated protection privileges.

In the CI or hierarchical file system, each directory has an owner and an associated group. All files in a directory are owned by the directory owner and have the same associated group as the directory. The owner can change the protection status (defined as read/write access allowed the owner, members of the associated group, and general users) of the files in the directory.

The session user who creates a directory is the initial owner of that directory and the directory has the owner's associated group. The owner can re-assign directory ownership and associated group. The owner and associated group of a subdirectory can be different from the owner and associated group of the directory containing the subdirectory.

File protection, a security measure in the CI file system, is defined when a file is created or copied into a directory. It can be specified differently for the owner, members of the associated group, and general users. The default protection is to allow the owner both read and write access, and associated group members and general users only read access. Protection can be specified on a file-by-file basis in any combination of read and write access for the owner, members of the associated group, and general users.

Directories also have protection information which is slightly different from that of files. The protection status of a directory applies to any file or subdirectory created in the directory. Users cannot use CI commands to change information in write-protected directories (read only access specified). They cannot create, purge, or rename files or subdirectories in the directory. Users cannot find out the contents of directories that have read protection.

In the FMGR file system, protection is defined by accessibility. FMGR cartridges (a particular LU defined at system generation) have a single directory and a single set of files.

Volume Ownership

Like a directory, an entire CI volume may have an owner and associated group. The initial owner of a volume is the session user initializing the volume. The associated group of volume is the associated group of the owner. Ownership and associated group may be reassigned. Volumes also have protection information. The protection status of a volume regulates the reading, creating, purging, or renaming of global directories on that volume. Note that the owner of a global directory can be different from the owner of the volume on which it resides.

In CI, any directory may access remaining volume space. If you want to restrict disc space usage, you can assign each CI volume an owner who will set volume-wide protection. Use the luV parameter in the CI OWNER and PROT commands to display and modify volume ownership and protection. You can also use FMP routines FmpOwner, FmpProtection, FmpSetOwner, and FmpSetProtection to retrieve and modify volume ownership and protection programmatically.

Security/1000 (VC+ Only)

Security/1000, part of VC+ software product (92078A), is designed to be compatible with existing protection, have minimal impact on real time processes, and be an integrated part of resource management. It is:

- Switchable - turned on and off with a single command.
- User configurable - users may tailor it to their own configuration by modifying the supplied template.
- User extensible - users can secure their own applications to create a unified security environment.
- Table-driven - facilitates user extensibility and configurability and allows modification while the system is running.

System Manager Responsibilities

If Security/1000 is used, your responsibilities as system manager include:

- Determining user, commands, and program capability levels within the multiuser environment.
- Installing the subsystem at system generation or regeneration.
- Using the SECTL utility to initialize and turn on Security/1000.
- Maintaining the security tables on-line with SECTL.
- Altering security tables off-line, running SECTL GT, and regenerating (to add more categories or category functions).

- Maintaining the subsystem using the GRUMP utility to alter user capability levels, and SECTL and LINK to modify program capability levels.

Categories, Functions and Security Tables

Security/1000 allows access to resources according to system manager-defined capability levels (CPLV) and security tables. See Appendix B for a description of the format of the security tables and for a sample of the source code of the SECURITY.TBL module that is shipped with the VC+ software.

A category is a group of functions such as the CI commands or the FMP routines. The category designer determines the group of functions.

A function is any activity carried out by the system on behalf of the the user. An individual function can be divided into four levels, the base function and up to three subfunctions. For example, the CI command TM displays the system time as its primary or base function. It also has the subfunction of setting the system clock. The function designer determines the base and subfunctions of any function.

The Category Index Table (CIT) and the Category Function Tables (CFT) are the security tables that determine access to system resources. See Figure 2-2 for the basic structure of these tables. The size of the CIT and CFTs are limited by available memory.

The CIT contains an entry for each category including its name and address. The CIT determines which CFT will be searched for a given category. The CFTs contain an entry for each function of a particular category. There is one CFT for each entry in the CIT. A CFT entry holds up to four integer values per function. These values are usually capability levels.

When Security/1000 checks to determine whether or not a function may be performed, it first searches the CIT for the address of the category's function table. Then it searches that particular CFT for capability information about the function and allows or disallows the function accordingly.

Searching the CIT and CFTs is done serially. If you have a category with a large number of functions, you will derive a performance advantage in having the functions divided between two smaller categories rather than having them in one large category.

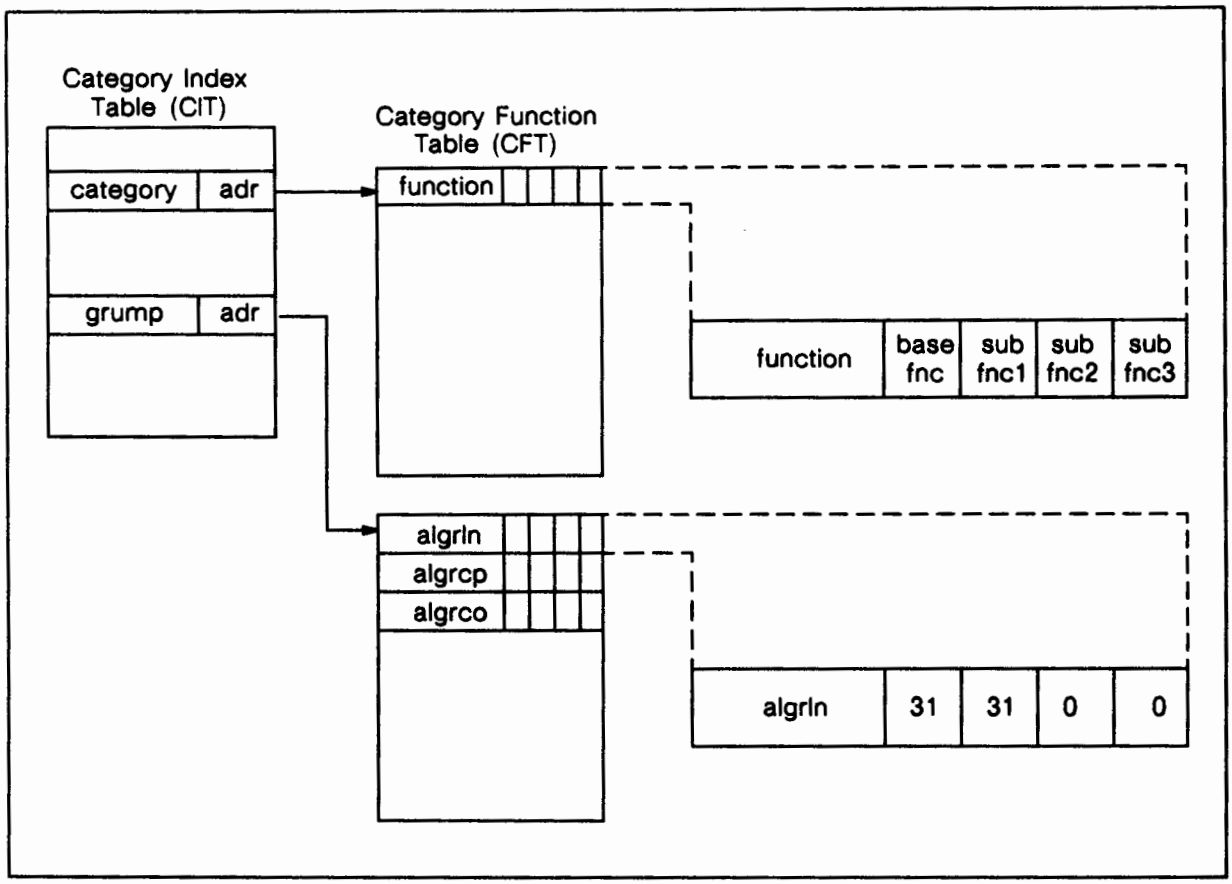

Figure 2-2. Security Table Structure

Security Information Example

Security tables hold primarily security information. In the example below, the function defined holds only security information.

```
$category: grump
              algrln 31 31 0 0
```

The category is the GRUMP utility for managing resources in the multiuser system. The function is defined as altering a group logon name. The base function, altering your own group logon name, requires a capability of 31. Subfunction 1, altering the group logon name for someone else, requires a capability of 31. Subfunctions 2 and 3 are not defined. Only capability levels are defined for this function.

Non-Security Information in Security Tables

A CFT can also hold non-security information. The interpretation of values depends upon their definition in the program that uses them.

The two methods of entering non-security information are:

1. Creating a new set of tables by:

   a. Altering a copy of the security table source file (the original SECURITY.TBL file or your already modified security table);

   b. Using the SECTL GT command (or STGEN) to generate a new set of tables;

   c. Regenerating the system; and

   d. Rebooting.

2. Modifying on-line security tables while the system is running by:

   a. Using SECTL commands to alter existing security tables, or

   b. Using the SEC1000.LIB subroutines (the programmatic interface of Security/1000) described in Appendix C to alter existing security tables.

Since you decide the definition of values, you can define non-security information as values. You can, for example, design an application that sends reports to different LUs.

```
$category: report
    rep01  8  0  0  0
    rep02 10 11 12 13
```

In this example, the integer 8 is defined as the LU to which REP01 is sent and integers 10, 11, 12, and 13 are defined as LU numbers to which REP02 is sent. Also, zero is used as a place holder. Note that the application processing the values is responsible for interpreting their meanings.

If a function requires more than four values, a way to work around the limitation of four values per function is to define two functions identically. Expanding upon the previous example,

```
$category: report
    rep01  8  0  0  0
    rep02 10 11 12 13
    rep2a 14  0  0  0
```

If the application has defined REP2A as identical to REP02, the integer 14 is interpreted as a fifth LU to which REP02 will be sent.

The rangeoff option is another example of the way in which function designers define values to accomplish their purpose. Use the rangeoff option when defining functions to specify values that are outside the 0 to 31 range. In the following example, the enable function of the application is set to 40, a value outside the 0 to 31 (inclusive) range:

```
$category: applic,rangeoff
    enable 40 0 0 0
```

The integer 40 is defined as the enabling value and any other value would disable the application. The base function is set to 40 for enable, non-40 for disable.

Security Table Format

The security tables are accessed via the $VCTR entry point $SEC.PNTR that points to the first word of the CIT. The first word of the CIT is the number of entries in the CIT. Each entry of the CIT has the following format:

Word	Contents
1-3	The category identifier. This is an up shifted, left justified and blank filled ASCII string that uniquely identifies the category to Security/1000. The category name can be any length but only the first six bytes are put in the entry. Therefore,the name must be unique within the first six bytes.
4	The address of the CFT for the category identified in words 1-3.

The first word of the CFT is the number of entries in the CFT. Each entry of the CFT has the following format:

Word	Contents
1-3	A function, left justified, up shifted and blank filled. The function can any length but only the first six bytes are put in the entry. Therefore, all functions for a given category must be unique within the first six bytes.
4	The Base CPLV for the function in the range 0-31 inclusive.
5	Subfunction 1 CPLV in the range 0-31 inclusive.
6	Subfunction 2 CPLV in the range 0-31 inclusive.
7	Subfunction 3 CPLV in the range 0-31 inclusive.

The values in words 4 through 7 can be greater than 31 if the function holds non-security information.

Capability Levels

A capability level (CPLV) is an integer in the 0 to 31 (inclusive) range. You, the system manager, can assign all users capability levels (USERCPLV), which are stored in the user configuration file and copied to the user ID table entry when the user is logged on.

You also can assign a capability level to all CI and system level commands (CMDCPLV). The CMDCPLVs for all commands are kept in security tables maintained by SECTL. If a command does not appear in a security table, it has a default capability of zero where no restrictions apply to the command. Note that a set of fully configured tables (file SECURITY.TBL) is supplied with the product and is shown in Appendix B.

All programs have three capability levels; two are accessible via SECTL and the Security/1000 routines and the third is for Security/1000 internal use. All three capability levels are stored in word 47 ($CPLV) of the program's ID segment. These CPLVs are:

PROGCPLV — Program capability level. PROGCPLV is the limiting factor on what functions a program can perform. PROGCPLV must be equal to or greater than function CPLV. PROGCPLV is accessible via SECTL and Security/1000 routines.

RQUSCPLV — Required user capability level. USERCPLV must be equal to or greater than RQUSCPLV in order to run the program. RQUSCPLV is accessible via SECTL and Security/1000 routines.

ORGCPLV — Original capability level. ORGCPLV is a copy of the original PROGCPLV given to the program at link time or set with the SECTL utility. ORGCPLV is for Security/1000 internal use; it is not directly accessible to users.

All fields of $CPLV have a default of zero.

You need to be aware of user, program, and system capability requirements as well as File System Security when determining capability levels. If the functions users need have been assigned CPLVs higher than their own, users will not be able to continue without some indirect means of accessing those programs (see the Program Access Protection section).

When a program is run with Security/1000 turned on, PROGCPLV is assigned the greater value of the session user's USERCPLV and the program's original PROGCPLV. Programs assigned a low CPLV can therefore perform tasks requiring a higher CPLV if the user requesting them has that higher CPLV. For example, if USERCPLV is 20 and PROGCPLV is 12, the system sets PROGCPLV to 20. Further, if PROGCPLV is 25 and USERCPLV is 12, PROGCPLV is set to 25, the greater value.

If a program is scheduled by another program, the USERCPLV of the session in which the father program is running is checked against the RQUSCPLV of the son program to determine whether the scheduling program can run the son program.

Program Access Protection

Because programs have three CPLVs, you can design applications whose capability is higher than that of the user. Users can be denied direct access to potentially dangerous functions, but be allowed access indirectly through the controlled environment of the application or utility.

There are two levels of checking before a user can run a program:

Level 1 File system security. A user must be able to access a program for it to be RPed. It must be RPed in order to be run.

Level 2 The USERCPLV of the session that makes the scheduled request must be greater than or equal to the RQUSCPLV of the program being scheduled. This check is made by the RTE-A Scheduler.

If a program does not have a ID segment, both levels of security checking must be passed to run the program. If the program has an ID segment and is RPed, only the second level must be passed because file system security is not involved with the schedule request.

If a user passes the first level and RPs the program but fails the second level and cannot schedule it, the operating system removes the ID segment and returns a message to the user's terminal. It is possible that a user able to run a program if it were RPed is prevented from running it because of file system security.

Programs in the system session are handled in the same way as programs in a user session. However, as the system session always has a USERCPLV of 31, programs running in it behave as though run by a superuser.

The two examples that follow illustrate program access within a secured environment.

Program Access Utility Example

Assume that a system has these user categories with these CPLVs:

General users	5
Junior operators	10
Senior operators	12
Application programmers	20
System programmers/System manager	31

Also, assume that the junior operator has the job of removing old files from a data base that receives new files daily (see Figure 2-3). The junior operator has a USERCPLV of 10. Because the Purge Data File function has a CPLV of 26, Security/1000 prevents the junior operator from removing the old files. Therefore, the junior operator's system manager created a utility, UTIL1, which allows specific, limited access to the Purge Data File function.

UTIL1 specifies that only files input at least two weeks prior to current system time may be removed. It has a RQUSCPLV of 10 so that the junior operator can access it, and a program CPLV of 26 so that it can access the purge function. UTIL1 affects only users with CPLVs of 10 through 25. Users with CPLVs below 10 are still denied access and users with CPLVs greater than 26 have full access to the purge function anyway. UTIL1 provides limited (CPLVs 10-25) and specific (purge only selected dated files) access.

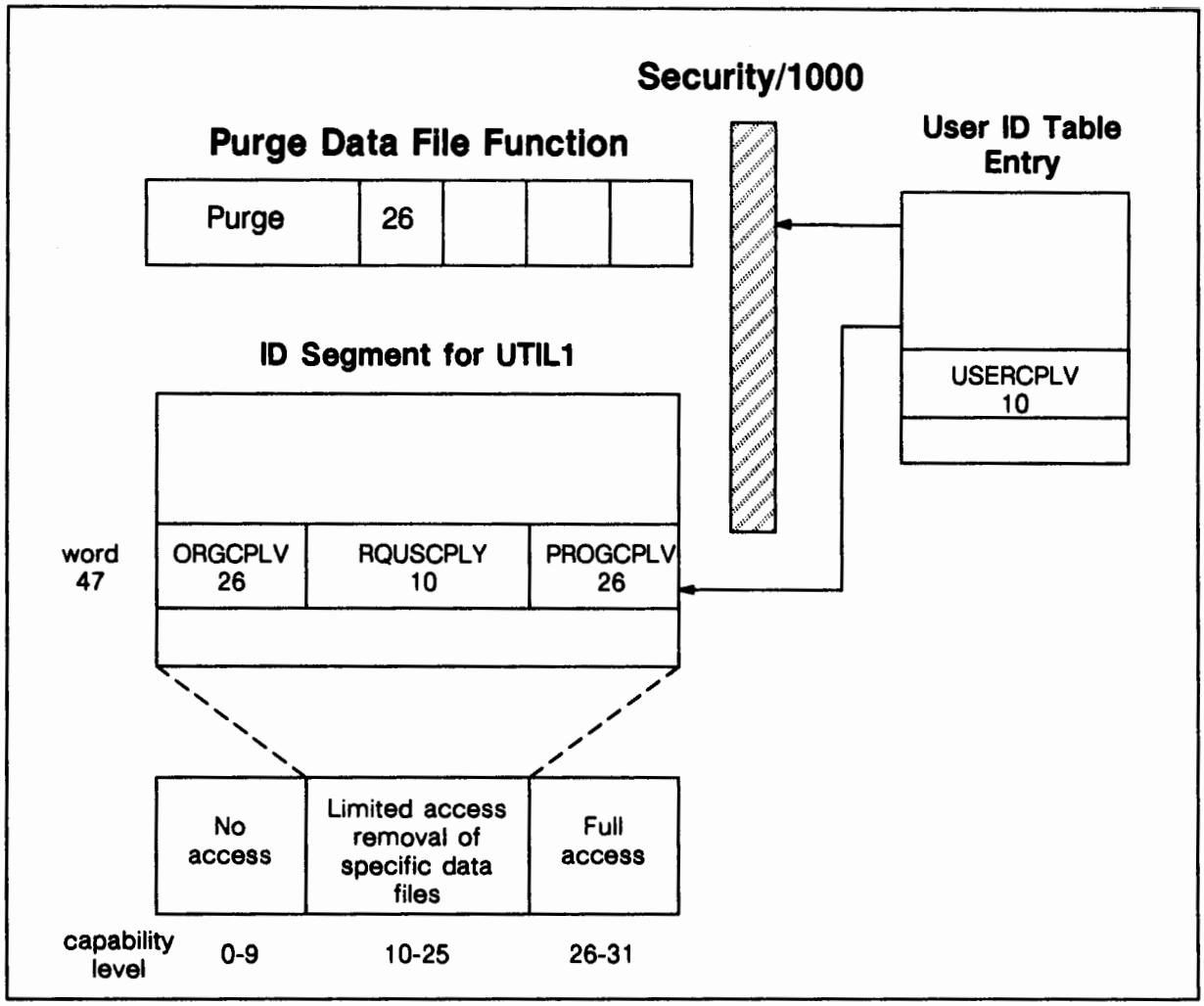

Figure 2-3. Example of Program Access Utility

CPLV Modification Program Example

Assume that it is the job of the senior operator, JAMALA, to do backups and restores via TF (see Figure 2-4). JAMALA has a USERCPLV of 12. Note that this example assumes that the system manager has used Security/1000 routines to modify the TF utility so that has different capability levels are needed to access different functions within TF.

At installation, the section of the security tables relating to TF would be as follows:

```
$fmp:
      .
      .
      .
      crdir 20 20 20 20
      .
      .
      .
$category: tf
*
* < Function cos is defined as storing files to tape
* < and the subfunctions define what copy options can be
* < used with the base function.
* < base fnc: who can store files to tape
* < subfnc01: who can use the verify option
* < subfnc02: who can use the purge option
* < subfnc03: who can use the append option
*
  cos 10 10 25 12
*
* < Function cor is defined as restoring files from tape
* < and the subfunctions define what copy options can be
* < used with the base function.
* < base fnc: who can restore files from tape
* < subfnc01: who can use the verify option
* < subfnc02: who can use the replace duplicate option
* < subfnc03: who can restore files to directories other than
* <           they originated.
*
  cor 12 10 12 31
*
$category: tfuser
           user01 0
             .
             .
             .
           user10 10
           user11 11
           user12 12
             .
             .
             .
           user31 31
*
```

TF was linked with a PROGCPLV of 20 and a RQUSCPLV of 10.

When JAMALA runs TF, the system sets PROGCPLV to 20, the greater value of PROGCPLV (20) and USERCPLV (12). This lets JAMALA restore files from a tape requiring that TF create directories even though the directory creation function has a CPLV of 20 (which is beyond JAMALA's CPLV).

TF was linked with a PROGCPLV of 20 but the security table listed above sets the CPLV of subfunction 2 of the cos function to 25. As a result only users with a USERCPLV equal to or greater than 25 can access the purge option.

When a junior operator such as MAX runs TF, the PROGCPLV is 20, the greater value of PROGCPLV (20) and USERCPLV (10). This is not a desired result because it gives MAX access to a function (cor) and an option (append of the cos function) to which a junior operator should not have access. MAX has access to the same base and options of these functions as JAMALA.

To avoid this situation, TF has a dynamic PROGCPLV modification facility implemented with Security/1000 routines and a security table category called TFUSER. This PROGCPLV modification facility causes TF to raise or lower its PROGCPLV during execution. It can lower its PROGCPLV to zero and raise it to the maximum of USERCPLV and ORGCPLV.

Now when MAX runs TF, the modification facility detects the USERCPLV of 10 and lowers its PROGCPLV to 10. When JAMALA runs TF, the modification facility changes the PROGCPLV to 12, MAX will not be allowed to use the restore function but JAMALA will. When JAMALA is performing a restore, TF will raise its PROGCPLV to 20 prior to the creation of the directory and lower it to 12 after the directory is created.

TFUSER is created with SECTL or Security/1000 routines if there are extra categories in the current security table (see the Installing Security/1000 section). If there are no extra categories in the current security table, you must create a new table which includes TFUSER and generate the tables in your system. These values are easier to change if need arises because they are in the security tables rather than hard coded into the program.

TF determines the correct value for its PROGCPLV by means of TFUSER which contains the values to which PROGCPLV is mapped for a given user. TF concatenates USER and the ASCII representation of the USERCPLV to get the function name (USER10 for MAX and USER12 for JAMALA in this example). It then calls routine SecGetCftNam to get the value to which its PROGCPLV should be mapped. Note that while the function value is equal to that of the USERCPLV in this example, it does not have to be.

TF calls routine SecChangeCplv to set its PROGCPLV to the correct value. Once the correct value for its PROGCPLV has been set, no special internal filtering by the program is required to restrict a low CPLV user to the desired functionality subset. The PROGCPLV modification facility causes TF to raise its PROGCPLV to the required level before attempting to create a directory for the restore operation. After the directory is created, it then resets PROGCPLV to the value obtained from TFUSER. All filtering is done by the Security/1000 routines.

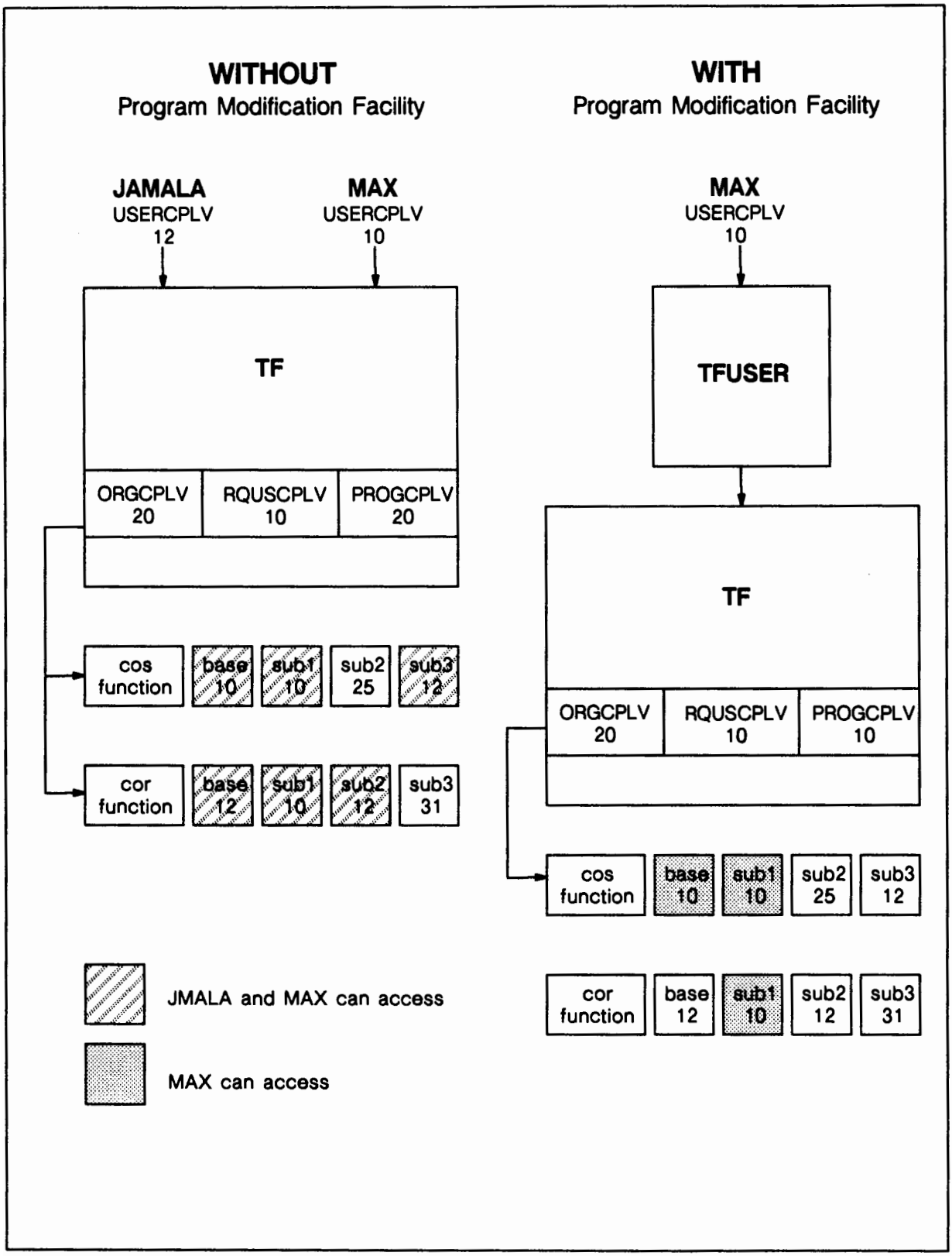

Figure 2-4. Example of a Program Modification Facility

Using Security Subroutines

See Appendix B for examples of two programs using security routines. The first program renames or edits the security tables. The second program subjects a user to a series of security checks based on the security tables before it will create directories. Programs such as CI, GRUMP, and LINK perform similar security checks on their commands.

Installing Security/1000

To install Security/1000, proceed as follows:

1. Load Security/1000 software (SECTL.LOD and STGEN.LOD) onto an RTE-A system with Revision 5000 or later.

2. Generate a set of tables using the SECTL GT command. See the GT command in the SECTL utility section in this chapter.

   A sample SECURITY.TBL (source code module) and SECURITY.REL are shipped with the product. If you do not want to alter the security tables and you do not want a SECURITY.MAC module, you do not need to generate a new set of tables. You can use the shipped SECURITY.REL in your system. A listing of SECURITY.REL is in Appendix B.

   If you do not want to alter the tables, but you want a SECURITY.MAC module, use the SECTL GT command (or STGEN) and SECURITY.TBL to create modules SECURITY.MAC and SECURITY.REL.

   If you want to alter the security tables to add categories or functions, you must alter a copy of SECURITY.TBL or your current security table source file. Use the SECTL GT command (or STGEN) and the modified security table source module to generate a new set of tables. Note that the other SECTL commands are used to modify and list categories and functions that have already been generated into a running system.

   To avoid having to generate tables to add categories or functions in the future, include blank modules in your initial table generation. These modules can later be modified with SECTL to accommodate new categories or functions. See the SECURITY.TBL source module for examples of reserving space for extra categories (HP000 and HP001), functions (FMP), and features (SECTL).

3. Put SECURITY into the system generation answer file.

   SECURITY.REL (or your equivalent) must be relocated in the System Message Block. Relocate the modules SECOS.REL and CHECK.REL (shipped on tape) with the other operating system module partitions. Both SECOS.REL and CHECK.REL are partitionable. See Appendix D for more information.

4. Reference the Security/1000 libraries in the Library List in your answer file.

   The Security/1000 library SEC1000.LIB (see Appendix C) must be put at the top of the non-CDS list. The non-CDS list should also have SEC1000.LIB at the end. SEC1000CDS.LIB must be at the top of the CDS list, followed by the rest of the CDS libraries, SEC1000.LIB, and ending with BIGLB.LIB. If you do not put the security libraries in the correct place, your programs will not load correctly. Undefined externals will result. This can be overcome by referencing the libraries in your list command files.

5. Generate the system and relink all your programs using the new system snap file.

   Be sure to relink SECTL and STGEN as directed in Step 1 with the new system snap file.

6. Enter the following line as the first line of the WELCOME file:

```
SECTL,+in:<snapfile>+on
```

7. Boot the new system.

Turning On Security/1000

Use the SECTL SW command or a SECTL runstring to turn on Security/1000.

It is the system manager's responsibility to turn on Security/1000 in a timely manner. You should not encounter problems if you turn on security at system boot because turn on will be done prior to any other activity on the system.

If, however, you turn on security at some time other than at the start of the WELCOME file, you may encounter problems. Figures 2-5 and 2-6 and the remainder of this section discuss situations of which you should be aware.

In a system with security off, word 47 of a program entry is ignored. That is, there is no check on the PROGCPLV, RQUSCPLV, or ORGCPLV when users wish to run a particular program such as PROG1.

When security is on, the CPLVs in word 47 are checked against the USERCPLV of the user session in which the request was made to determine whether or not a scheduling program may run the son program.

In the example shown in Figure 2-5, the user has a CPLV of 10. When it was linked, PROG1 assumed the PROGCPLV, RQUSCPLV, and ORGCPLV defaults of zero because the user specified no other CPLVs. Since PROGCPLV is set to the higher value of the program's original PROGCPLV and USERCPLV, PROGCPLV is set to 10 when the user tries to run PROG1. USERCPLV must be equal to or greater than RQUSCPLV as it is here. So this user is allowed to run PROG1. PROG1, however, may not be able to do much because of its low CPLV of 10. The user is limited to the capabilities provided in that environment which, may be your intention.

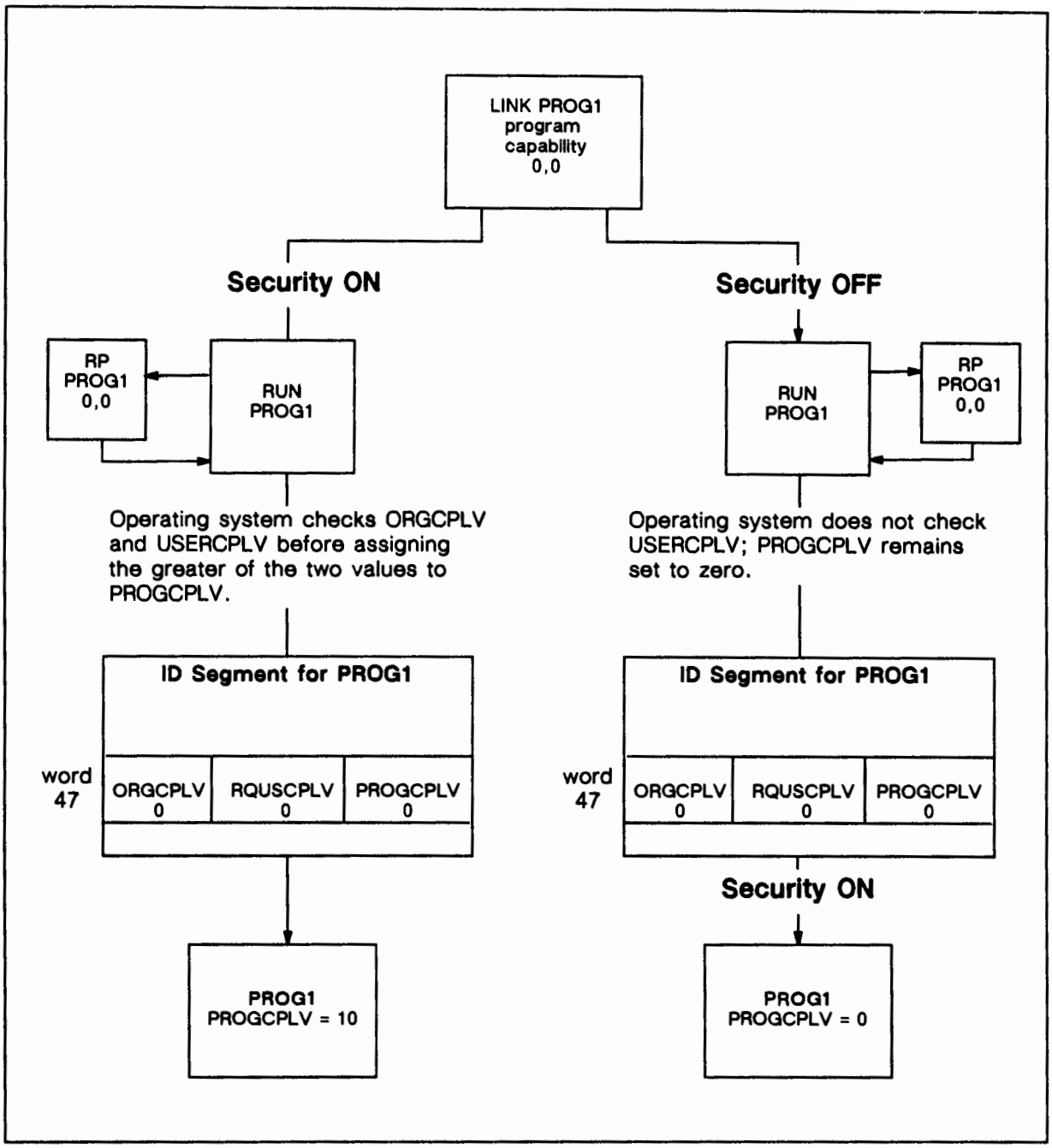

Figure 2-5. Turning On Security/1000

A way to visualize what can happen when security is turned on at a time other than at bootup is to imagine the DS subsystem linked and scheduled with a CPLV of zero before security turn on. Several users are using DS lines at their terminals. Their programs were linked with default CPLVs of zero because no other CPLVs were specified. Since the programs were scheduled before security was turned on, their PROGCPLVs were not set to the maximum of PROGCPLV and USERCPLV and still have PROGCPLVs of zero. Security is turned on at this point. The DS program becomes unusable, programs abort, and users cannot do anything.

The point to remember is that programs in use before Security/1000 is turned on will probably not be able to run if Security/100 is turned on and they were not linked with a high enough capability level. Of course, if they were linked with a CPLV of 31, there would not be the same problem.

You should be aware of another situation which can occur with programs linked before security is turned on. Users can link programs with any RQUSCPLV and any PROGCPLV when Security/1000 is off. As shown in Figure 2-6, users with a USERCPLV of 20 are linking a program, PROG1, with a RQUSCPLV of 10 and a PROGCPLV of 31.

If security is on when these users link PROG1, they receive the message that they have insufficient capability to use the PC command. PROG1 gets an ORGCPLV and PROGCPLV of zero and a RQUSCPLV of 10.

If security is off when these users link PROG1, security checks are not performed and PROG1 gets an ORGCPLV and PROGCPLV of 31 and a RQUSCPLV of 10. When security is turned on later, users running these programs have access to functions they should not. Users with USERCPLV of 10 or more can use PROG1 with its PROGCPLV of 31. If they have access to the LINK PC command when security is off, you may want to purge users' individual programs before turning on security and have the users relink them after security is on.

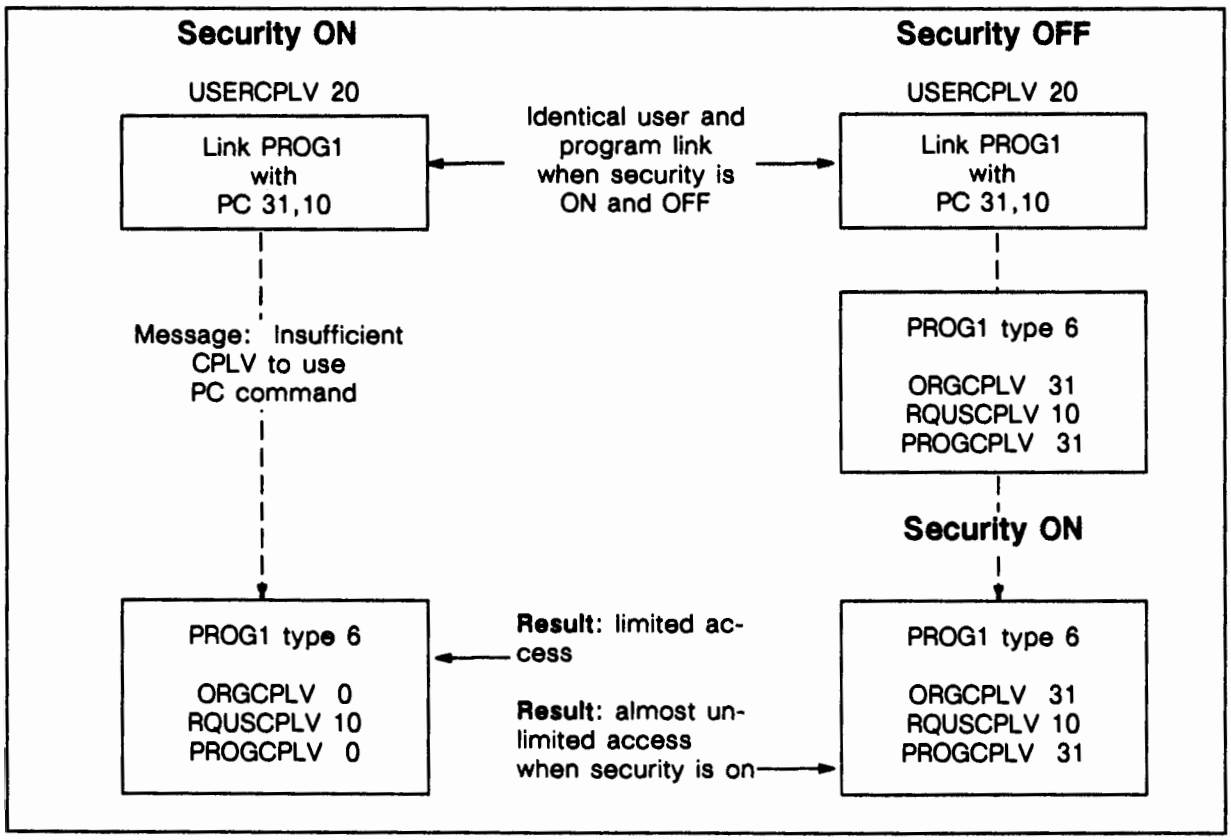

Figure 2-6. Linking Programs and Turning On Security/1000

SECTL Utility

Security/1000 has two interfaces, the SECTL utility which is the user or interactive interface, and the SEC1000.LIB subroutine library which is the programmatic interface, SEC1000.LIB is in Appendix C.

The SECTL utility performs all the control functions of Security/1000:

- Initialize Security/1000
- Turn Security/1000 on or off
- Set or modify the program capability levels
- Generate a set of security tables
- List the security tables to a device or file
- Edit the security tables
- Display the Security/1000 help file

Remember that commands issued to SECTL are subject to security checking and only authorized users (those with enough capability) can control Security/1000 operation.

Changes which are made with SECTL are not permanent. When you reboot, the values return to what they were when the system was generated. You must modify the source module for the security table, generate a new system and reboot for permanent changes.

Running SECTL

The SECTL runstring has the following three forms:

```
[RU,]SECTL[,infile[,outfile]]

[RU,]SECTL[,+IN[:snapfile]][,+ON]

[RU,]SECTL,+OFF
```

infile	File from which SECTL reads its commands; defaults to the terminal; if it is a disc file, it must have a "SEC" extension.
outfile	File to which SECTL sends its output; defaults to the terminal.
+IN	SECTL initializes Security/1000.
snapfile	Name of current system snapshot file; default is /SYSTEM/SNAP.SNP.

+ON	Turns on Security/1000; can be used only if Security/1000 has been initialized.
+OFF	Turns off Security/1000; can be used only if Security/1000 hasbeen initialized.

Description:

If the plus options are used in the run string, SECTL will terminate as soon as they have been processed.

SECTL Command Summary

SECTL commands are summarized in Table 2-1.

Table 2-1. SECTL Commands

Command	Purpose
EC	Modify category function capability
EX	Terminate SECTL
GT	Generate security tables
HE	Display Security/1000 help file
IN	Initialize Security/1000
LT	List Security tables
PC	Set program capability
RQ	Set required user capability
RN,C	Rename category ID
RN,F	Rename function ID
SW	Turn on/off Security/1000

Edit Capability Level (EC)

Purpose: Changes the CPLVS for a given category function.

Syntax: `EC,cat-id,fnc,base-cplv[,sub-cplv1[,sub-cplv2[,sub-cplv3]]]`

cat-id	Category defined in currently installed CIT of the security tables
fnc	Function defined in CFT belonging to the specified category ID
base-cplv	Base function's capability level
sub-cplv1	Subfunction 1's capability level
sub-cplv2	Subfunction 2's capability level
sub-cplv3	Subfunction 3's capability level

Description:

An example of an EC command is:

```
ec,grump,algrln,20,25
```

In this example, the ALGRLN function in the GRUMP category alters the group logon name. The base function, which defines the CPLV required to change your own group logon name, has been changed from 31 to 20. Subfunction 1, which defines the CPLV required to change the group logon name for someone else, has been changed from 31 to 25. The values for the second and third subfunction are not changed.

If you want to change the CPLV for the third subfunction and leave the other CPLVs as they are, you can use commas as place holders. The other CPLVs default to their current value. An example of this is:

```
ec,link,pr,,,,25
```

The CPLV of the base function and the first and second subfunctions have not been changed. The CPLV of the third subfunction has been changed to 25.

Exit (EX)

Purpose: Terminates SECTL.

Syntax: `EX`

Generate Table (GT)

Purpose: Generates a set of security tables.

Syntax: `GT,source,listfile,relfile[,keepmac]`

source	Contains the source code of the table to be generated
listfile	Destination of listing
relfile	Contains generated tables
keepmac	Contains MACRO/1000 generated as intermediate step of security table generation

Description:

Some examples of the GT command are:

```
gt,security.tbl,-,-,-
```

This example accepts default SECURITY.LST, SECURITY.REL and SECURITY.MAC files. Dashes must be used as place holders.

```
gt,security.tbl,tom.lst,dick.rel
```

This example sends the listing to TOM.LST, the generated tables to DICK.REL, and does not create a Macro file.

```
gt,security.tbl,0,-
```

This example creates no list file, creates the default relfile, and does not create a Macro file.

Help (HE or ?)

Purpose: Displays the Security/1000 help file.

Syntax: `HE`

`HELP`

`?[?]`

Initialize (IN)

Purpose: Initializes Security/1000 using the specified snap file.

Syntax: `IN[,snapfile][,ON]`

snapfile	File used to initialize Security/1000. If not specified, the default is /SYSTEM/SNAP.SNP.
ON	Turns on Security/1000 after initialization. If not specified, Security/1000 is initialized but not turned on.

Description:

Security/1000 must be initialized before it can be turned on.

An example of the IN command is:

```
in,/system/snap.snp,on
```

List Table (LT)

Purpose: Lists the currently installed security tables.

Syntax: `LT[,listfile]`

listfile	Destination file for the list of security tables. If the file exists, it is overwritten. If the file does not exist, it is created. If not specified, the default is LU 1.

Description:

Examples of the LT command are:

```
lt,tom.lst

lt,jamala.lst
```

Program Capability (PC)

Purpose: Sets the program capability level.

Syntax: `PC,program,cplv`

program	Program whose capability level in the ID segment is to be reset.
cplv	New capability level.

Description:

The value of the PROGCPLV cannot exceed that of the USERCPLV for the person running SECTL. The program must have an ID segment which will be updated. The program file will not be updated with the PC command. You must use LINK to update the program file.

An example of the PC command is:

```
pc,setup,31
```

In this example, the PC command sets the PROGCPLV for the SETUP program to 31. Since the PROGCPLV defined cannot be greater than the USERCPLV of the person using SECTL, the user in this example must have a USERCPLV of 31.

Rename a Category (RN,C)

Purpose: Renames a category ID in the category index table (CIT).

Syntax: `RN,C,old-cat-id,new-cat-id`

old-cat-id	Old category ID to be renamed in the CIT.
new-cat-id	New category ID in CIT.

Description:

The old category ID must exist in the CIT and the new category must not.

In the following example, category CAT1 in the CIT is renamed to CAT2:

```
rn,c,cat1,cat2
```

Do not rename standard categories such as CI or all security becomes ineffective.

Rename a Function (RN,F)

Purpose: Renames a function in a CFT

Syntax: `RN,F,cat-id,old-fnc-id,new-fnc-id`

cat-id	ID of the category containing the function to be renamed.
old-fnc-id	Function in the CFT to be renamed.
new-fnc-id	New function ID in the CFT.

Description:

The old function ID must exist in the CFT and the new function ID must not.

In the following example, the CMDL function in the CAT1 category is renamed to CMD1:

```
rn,f,cat1,cmdl,cmd1
```

You should not rename standard functions because all security becomes ineffective.

Required User Capability (RQ)

Purpose: Sets the program's required user's capability level (RQUSCPLV).

Syntax: `RQ,program,cplv`

program	Program whose RQUSCPLV is to be set.
cplv	New RQUSCPLV.

Description:

The value of RQUSCPLV cannot exceed the USERCPLV of the user running SECTL. The program must have an ID segment that will be updated. The program file will not be updated with the RQ command. You must use LINK if you want to update the program file.

In the example,

```
rq,setup,28
```

the RQUSCPLV of the SETUP program is set to 28. Because, the RQUSCPLV of a program cannot be greater than the USERCPLV of the person running SECTL, the user must have a USERCPLV of at least 28.

Switch (SW)

Purpose: switches Security/1000 on or off.

Syntax: `SW,ON|OFF`

ON	Switches on Security/1000.
OFF	Switches off Security/1000.

Description:

This command can be used only after Security/1000 has been initialized.

*

Purpose: An * in column 1 means the rest of the line is a comment. This is used after SECTL commands are placed in a command file.

Description:

An example of the use of the asterisk (*) is as follows:

```
* The following commands create
* category CRDGP from extra category
* HP000.
    rn,c,hp000,crdgp
    rn,f,crdgp,res00,nogroup
    rn,f,crdgp,res01,system
    ec,crdgp,system,0,1,2,3
    ex
* End of creation.
*
```

STGEN

When installing Security/1000, you can generate security tables directly by running the STGEN program rather than using the SECTL GT command.

Using STGEN is not generally recommended for generating tables. However, you can use it if you want to generate tables from a CI command file and want to know if there are any errors and what those errors are. STGEN returns all error codes to CI in the $RETURN1 through $RETURN5 variables.

The STGEN runstring is as follows:

```
[RU,]STGEN,source,list,relocatable,keep
```

source	The name of the file containing the source definition of the security tables.
list	The name of the list file. <source>.LST is the list file if it is a dash (-). Use zero if you do not want a listing.
relocatable	The name of the file that will contain the compiled security tables. <source>.REL will be the name of the relocatable file if a dash (-) is used. Use zero if you do not want a relocatable file.
keep	The name of the file where the Macro/1000 that is generated will be kept. <source>.MAC will be the name of the keep file if a dash (-) is used. Use zero or a blank if you do not want a keep file.

Security Table Source Example

This example shows the files generated from the source code module using SECTL with the GT command.

An asterisk in the first column means the text in that line is a comment. Blanks lines are allowed.

```
*-----<  The security category for GRUMP.
*
*
*-----<  Meanings of the base and subfunctions are:
*-----<
*-----<  (1)Base  -  Capability needed to perform the function on yourself.
*-----<     Sub01 -  Capability needed to perform the function on
*-----<              someone else
*-----<     Sub02 -  Not defined
*-----<     Sub03 -  Not defined
*-----<
*-----<  (2)Base  -  Capability needed to perform the function on a group
*-----<              in which you are a member
*-----<     Sub01 -  Capability needed to perform the function on a group
*-----<                 in which you are not a member
*-----<     Sub02 -  Not defined
*-----<     Sub03 -  Not defined
*
* $category: grump

   algrln  31  31  0   0   *alter group logon name                (2)
   algrcp  25  31  0   0   *alter group cpu limit                 (2)
   algrco  25  31  0   0   *alter group connect time limit        (2)
   algrbm  25  31  0   0   *alter group bit map                   (2)
   alusln  31  31  0   0   *alter logon user name                 (1)
   alusrn  10  31  0   0   *alter logon user real name            (1)
   aluspw  10  31  0   0   *alter user password                   (1)
   alusud  25  31  0   0   *alter user udsp depth and levels      (1)
   aluscl  31  31  0   0   *alter user cplv                       (1)
   alusbm  25  31  0   0   *alter user bit map                    (1)
   aluscp  31  31  0   0   *alter user cpu limit                  (1)
   alusco  31  31  0   0   *alter user connect limit              (1)
   alusgr  25  31  0   0   *alter user group list (ie add groups) (1)
   aludlg  15  31  0   0   *alter user default logon group        (1)
   alussu  15  31  0   0   *alter user startup program            (1)
   aluslf  15  31  0   0   *alter user logoff program             (1)
   aluswd  31  31  0   0   *alter user logon working directory    (1)
   ligrp   15  31  0   0   *list group definitions                (2)
   liusr   15  31  0   0   *list user definitions                 (1)
   passwd   0  25  0   0   *change user password                  (1)*
*-----<  The meanings of the base and subfunctions are now.
*-----<
*-----<  Base  - Capability needed to issue the command.
*-----<  Sub01 - Not defined.
*-----<  Sub02 - Not defined.
*-----<  Sub03 - Not defined.
*
            negrp   31  0   0   0   *create new group
            neusr   31  0   0   0   *create new user
            pugrp   31  0   0   0   *purge a group
            puusr   31  0   0   0   *purge a user
            regrp   31  0   0   0   *reset a groups accounting info
            reusr   31  0   0   0   *reset a users accounting info
            kilses  31  0   0   0   *kill a session
*
```

Example of Generated Macro/1000 Code

This is a sample of the Macro/1000 that is generated by the SECTL utility when generating a set of security tables.

```
macro,l,c
                hed SECURITY/1000 Table

*
*------<Source file:         SECURITY.TBL
*------<List file:           SECURITY.lst
*------<Relocatable file:    SECURITY.rel:::5
*------<keep file:           SECURITY.mac
*

                nam sctbl,0  92078-1X102  Rev.5000 861016.225844
                ent CatIndex,$sysct.adr,$exect.adr
                ent $secct.adr,$fmpct.adr
GRUMP.adr       dec 24
                asc 3,ALGRLN
                dec 31
                dec 31
                dec 0
                dec 0
                asc 3,ALGRCP
                dec 25
                dec 31
                dec 0
                dec 0
                asc 3,ALGRCO
                dec 25
                dec 31
                dec 0
                dec 0
                asc 3,ALGRBM
                dec 25
                dec 31
                dec 0
                dec 0
                asc 3,ALUSLN
                dec 31
                dec 31
                dec 0
                dec 0
                asc 3,ALUSRN
                dec 10
                dec 31
                dec 0
                dec 0
                asc 3,ALUSPW
                dec 10
                dec 31
                dec 0
                dec 0
                asc 3,ALUSUD
                dec 25
                dec 31
                dec 0
                dec 0
```

```
asc 3,ALUSCL
dec 31
dec 31
dec 0
dec 0
asc 3,ALUSBM
dec 25
dec 31
dec 0
dec 0
asc 3,ALUSCP
dec 31
dec 31
dec 0
dec 0
asc 3,ALUSCO
dec 31
dec 31
dec 0
dec 0
asc 3,ALUSSU
dec 15
dec 31
dec 0
dec 0
asc 3,ALUSLF
dec 15
dec 31
dec 0
dec 0
asc 3,ALUSWD
dec 31
dec 31
dec 0
dec 0
asc 3,LIGRP
dec 15
dec 31
dec 0
dec 0
asc 3,LIUSR
dec 15
dec 31
dec 0
dec 0
asc 3,NEGRP
dec 31
dec 0
dec 0
dec 0
asc 3,NEUSR
dec 31
dec 0
dec 0
dec 0
asc 3,PUGRP
dec 31
dec 0
dec 0
```

```
dec 0
asc 3,PUUSR
dec 31
dec 0
dec 0
dec 0
asc 3,REGRP
dec 31
dec 0
dec 0
dec 0
asc 3,REUSR
dec 31
dec 0
dec 0
dec 0
asc 3,KILSES
dec 31
dec 0
dec 0
dec 0
```

3

Multiuser Account System

This chapter discusses the multiuser account system. You create and maintain the multiuser account system with the GRoup and User Manager Program (GRUMP) described in detail at the end of this chapter. On-line help for GRUMP is provided.

To prepare for setting up the multiuser account system, you must perform the following tasks:

- Organize the user base into a hierarchy of groups and users. Groups should include sets of users with common characteristics or requirements such as members of a project team or individuals with similar functions. Users can be members of more than one group.
- Estimate the number and size of CI file volumes and FMGR disc cartridges in the system. This will depend on your account structure, user application requirements, and the degree of file independence required by various users of the system.

This information can be gathered in interviews with your users. See Design and Planning section in the System Management chapter for a sample questionnaire for determining user requirements.

The Session Environment

The process of logging on, interacting with the system, and logging off is referred to as a session. A user accesses the system, by supplying, at a minimum, a user name. A group name is optional. If the user wants to associate the session with a group other than the defined default group, then that group name must be specified. If the user has a password defined, it must be supplied at logon.

The system sets up an operating environment for each user session based upon the user and group accounts with which the session is associated. User and group account information is kept in user and group configuration files on the /USERS directory. Both user and group configuration files are created and modified with the GRoup and User Management Program (GRUMP).

If the user's startup program is CI, the session's associated user and group name are put in the predefined variable $LOGON in the form USER.GROUP.

After logon, the system permits only those user peripheral access requests and commands allowed within the operating environment. Users can access many peripherals with default logical unit numbers. They do not need to know system logical unit assignments. For example, each user's terminal is referred to as LU 1 rather than by the actual system LU assigned to it.

When done interacting with the system, the user logs off. If a user logs off via CI and a logoff program/command file has been defined, it is scheduled or transferred to. The system releases system resources allocated for the session and LOGON updates the user and group configuration files associated with the session. The user's total CPU usage and connect time for the session are added to the user's grand total, group's totals, and the totals for the user in that group and printed to the terminal if the user was in an interactive session.

Session Log On Process

LOGON, the session logon and logoff processor, operates in conjunction with PROMT and CM (an RPed copy of CI). It is invoked by PROMT when there is an interrupt at a terminal and no active session is operating from the terminal. At logon, LOGON prompts the user for logon entry (USER.GROUP name) and password (if required). LOGON attempts to match the logon entry with an existing USER.GROUP definition. If it finds a match, a user ID entry is created for the session and the user's session is initiated.

LOGON uses the group configuration file to

- Verify that the group exists
- Verify the user is a member of the group
- Check the group accounting limits
- Include the group's resources in the operating environment of the user's session

LOGON uses the user configuration file to

- Verify the user logon name
- Check the user's CPU usage and connect time limits within the group
- Run the start-up program
- Designate a working directory
- Initialize UDSP tables
- Create the session LU access table
- Set the session user's capability level
- Record the last logon time
- Record the LU the user logged onto
- Record the group ID logged onto by the user

LOGON creates a user ID table entry for each user session. The ID table entry contains information about the user session initially obtained from the user logon entry and later from the user and group configuration files set up in the system. The user ID table entry contains:

- User logon name
- Session sequence number
- Pointer to working directory
- Terminal LU of the user or session number
- Logoff program/command file bit
- Number of user programs counter
- User identification number
- Logon time for user logged on
- Session CPU usage
- Address of UDSP table in XSAM
- ID segment address of the first session program
- Group ID number
- User capability level

User-Definable Directory Search Path (UDSP)

There is a User Definable Directory Search Path (UDSP) associated with each session. A UDSP is a list specifying which directories to search when opening a file, and the order in which they are to be searched. There can be up to eight UDSPs, numbered from 0 through 8. UDSP 0 is a special case, it represents the 'Home' directory and has a depth of one.

The number of UDSPs and their depth (the number of entries per UDSP) are defined when the user account is created or modified. LOGON allocates space for the session UDSP in XSAM and initializes them when a user logs on. PATH is the utility that defines and displays UDSPs. The UDSP entries set by PATH are valid only for the duration of the session, they are initialized to undefined each time a user logs on.

The RU command in CI uses UDSP #1, and the TR command uses UDSP #2. Other UDSPs may be defined by the user for special use. Programmatically, D.RTR can be directed to use a UDSP when opening a file by specifying the UDSP number in the option string in an FmpOpen call.

You need to determine the optimal number and depth of paths for each user. The amount of XSAM is affected by the number of users that have UDSPs. Use the GRoup and User Management Program (GRUMP) to define number and depth of UDSPs. Command files can set up paths at logon.

Session LU Access Table

Each user and group definition contains a 16 word LU access table which provides a means of limiting access to LUs. If the LU number is in the LU access table, access to the LU is granted. If the LU number is not in the table, all access to the LU is denied. User and group LU access tables are created and modified with the GRoup and User Management Program (GRUMP).

NOTE

LU 0, the bit bucket, is used for program to program communication and should not be removed from the LU bit map. Also make sure that the LUs corresponding to all directories users need to access (such as /PROGRAMS) are in the LU access table.

LOGON creates the session LU access table by combining the LUs the user and associated group can access when a user logs on. The session LU access table is stored in XSAM. For technical detail about logon limitations when the system runs out of XSAM, see the RTE-A System Design Manual.

When session users try to access an LU, their LU access tables are checked. The check is done in the operating system after the actual system LU of the LU specified in the user's I/O request has been determined. If users have access to the LU, their requests are granted, otherwise, the requests are aborted. This check catches all I/0 requests no matter how they are issued (with FMP calls or EXEC calls) and no matter how they are re-directed.

Session Log Off Process

When a user has completed a session and logs off, LOGON, the logoff processor, updates the session account file with the CPU usage and connect time and de-allocates system resources for the session.

Session Utilities

To help you manage user sessions, the SESLU and KILLSES utilities are provided. A brief description of each utility follows; however, see the RTE–A Utilities Manual for detailed information on these utilities.

Modifying and Listing Session LU Access Tables

The utility SESLU lists and modifies session LU access tables. It does not affect the user or group configuration files associated with the session user. (See RTE-A Utilities Manual for information on SESLU.)

A user is not allowed to remove access to a session user's terminal LU or to any LU containing the current working directory or any directories specified in the UDSP table.

If Security/1000 is turned on, the system manager can assign different required capability levels for the base function and each subfunction of the SESLU utility. Users must have the required capability to run SESLU. See the section on Security/1000 for a detailed description of capability levels, base functions, and subfunctions. If the Security/1000 system is not turned on, any user can list a session LU bit map with SESLU, but a user must be a superuser to modify a session LU bit map.

Terminating a Session

KILLSES is a utility that terminates a session immediately. The user is logged off, all programs associated with the session are terminated, associated spool files are closed and released, and the user entry in the user ID table for the session is released. The session can be any type: background, interactive, or programmatic. The system session, session 0, is and exception and can never be killed. (See RTE-A Utilities Manual for details on using KILLSES.)

If Security/1000 is turned on, the system manager can assign the capability level required to run KILLSES. If Security/1000 is not turned on, the user must be a superuser to run KILLSES.

Account Structure

The multiuser account system maintains two types of accounts: user accounts and group accounts. A group is a set of users who share common functions, applications, and/or resources. Group accounts are used to assign selected resources to specific sets of users and to track accounting use for the groups. User accounts provide the system with the information necessary to set up and maintain the operating environment and track accounting use for that user.

Every session user must be assigned a user account. The user account contains unique user information and information about the user for each group in which the user is a member. Unique user information pertains to the user no matter what group the user logs on with. The information about the user in each group is called USER.GROUP information and pertains to the user only when logged on associated with the group for which the information is defined. Unique user information ensures that the same set of private resources are retained in the user's operating environment regardless of associated logon group. USER.GROUP information allows tailoring of the user's environment to the application needed for the associated logon group. It also allows for finer control of accounting.

All accounts are specified to the system in the form USER.GROUP where USER and GROUP are identifiers of one to sixteen characters in length. User identifiers and group identifiers must be unique in the system.

A user can belong to several groups, but for accounting is considered a separate entity in each. Example: Fred, logged on in group ABA, cannot switch to group CDC in mid-session. He must log off ABA before logging onto CDC.

An example account structure is shown in Figure 3-1. As can be seen from the example, the account structure is broken down into three levels: system manager, group, and user. Note in the diagram that Jones is a member of three groups so that Jones can access the same private files and/or peripherals from all three groups.

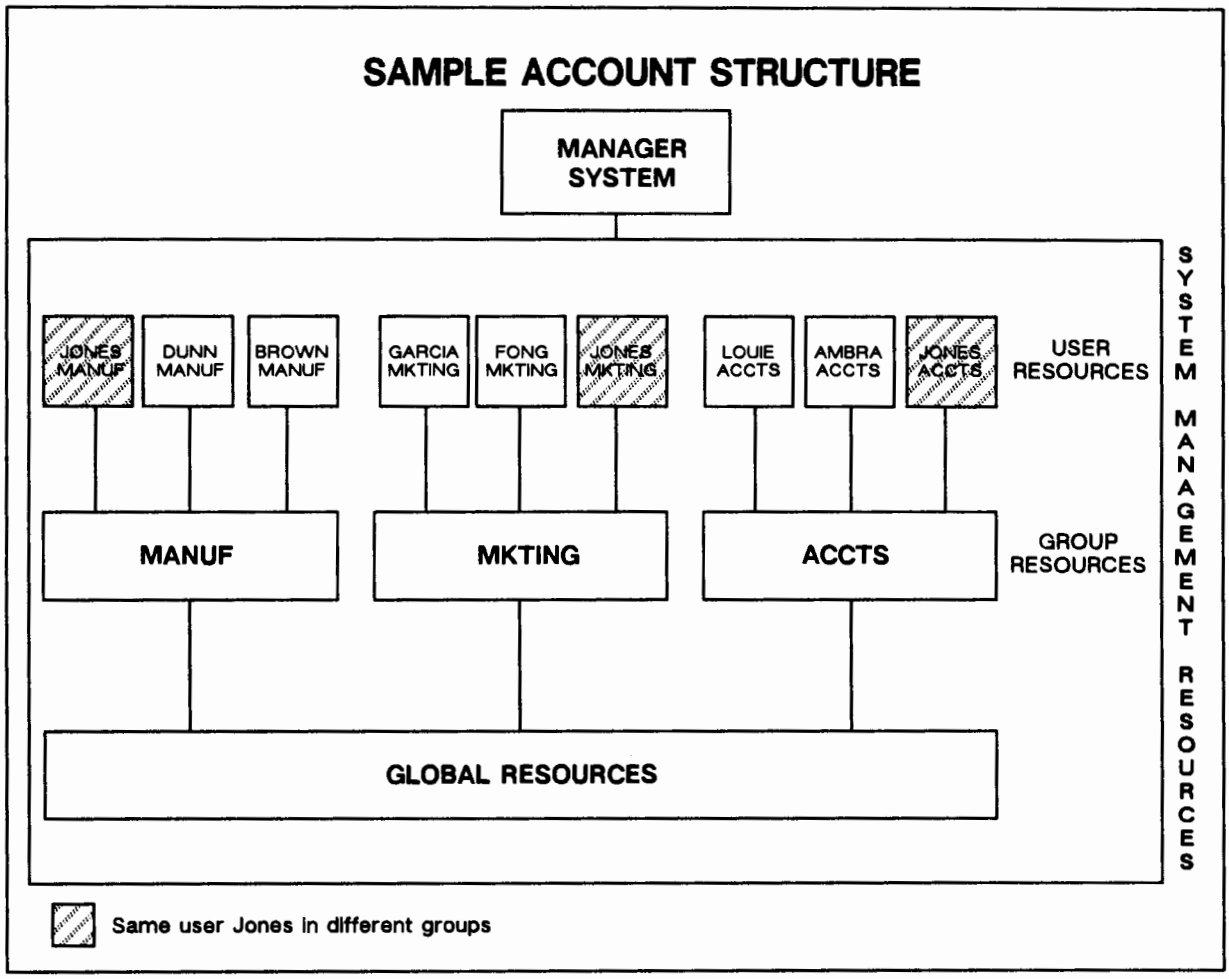

Figure 3-1. Sample Account Structure

User Account Planning

Use an account planning worksheet to list all the individual users of your system. For planning convenience, you should assign a unique identifier (up to 16 characters) to each user. A sample account matrix is shown in Figure 3-2.

GROUPS / USERS	MANUF	MKTING	ACCTS
JONES	X	X	X
DUNN	X		
BROWN	X		
GARCIA		X	
FONG		X	
LOUIE			X
AMBRA			X

Figure 3-2. Sample Account Planning Matrix

Group Account Planning

Once you have listed your system users, divide them into groups. Members of a group will usually share one or more common attributes. Some of the criteria that may apply here are explained below.

Existing Organization

You may find it convenient to follow an existing organizational pattern. Your account structure could reflect the actual groups in your user community.

Common Files

Users who share files or data bases can be included in a group. FMGR and disc LUs can be associated with a group in such a way that they can be accessed solely by members of the group. CI volumes and directories can be assigned protection in a way that allows group members to access them.

Common Peripherals

Groups can be formed around special peripheral access requirements. If desired, peripherals can be restricted to selected groups and/or users. By including the corresponding LU in the group's LU access table, peripherals may be defined to the account system so that they are automatically added to the list of peripherals individual group members may access.

Common Applications

You can separate users into groups based on their applications and/or job functions. Users performing similar tasks could then share related files and peripherals.

Usually, you should only form new groups when the list of users sharing a common resource is composed of users from two or more existing groups. If the users are all members of one existing group, you may be able to add the resource to that group's domain. The information gathered here will be used later on to initialize and maintain the account system.

Assign a name or an identifier, up to sixteen characters, to each group in your user community. This identifier must be unique. It will be used by members of that group to identify themselves to the system. List each group in the group column of the account planning matrix, see Figure 3-2. Next, indicate the members of each group. In each group column, place a check mark in all rows corresponding to the members of that particular group. Note that there is no restriction on the number of groups to which a user may belong. This may be a requirement in situations where individuals need to access resources owned by many different groups.

User Configuration File

The operating system maintains information on all user accounts in files called user configuration files. All user configuration files reside on the /USERS directory (which should be write protected for security) and each file name corresponds to a user's logon name. Each user configuration file contains information unique to the user regardless of associated group, and information unique to the user within each group in which that user is a member.

CAUTION

Do not purge a user configuration file from the /USERS directory with the PURGE command in CI. Use the PURGE USER command in GRUMP which purges the user from the multiuser account system with all the necessary cleanup.

The information in the configuration file is filled in during the creation of a new user account with the GRoup and User Management Program (GRUMP). The system manager or a user with the required capability supplies some of the information and GRUMP generates the rest. GRUMP also is used to modify or list information in the user configuration files.

Routines GetAcctInfo, ResetAcctInfo, and SetAcctLimits can be used to access and programmatically alter some of the user information. See the Relocatable Libraries Reference Manual (part no. 92077-90037).

The system uses the user configuration file name to verify the user's logon name. The unique user information in the file is used by the system to verify the user's password, initialize UDSP tables, update the last logon time, update the group ID that the user last logged on with, update the LU the user last logged on to, create the session LU access table, determine the capability of the user, update the total CPU and connect time usage for the user, and update the last logoff time.

The information unique to the user within a group is used by the system to check that the user's CPU usage and connect time limits within the group have not been exceeded, run the start-up program, designate a working directory, run the logoff program/command file, and update the CPU and connect time usage for the user within the group.

The information in the user configuration file falls into the following two categories:

1. *Unique User Information*: information associated with the user regardless of group.

   - Real Name
   - Superuser Bit
   - Encoded Password
   - User ID Number
   - UDSP Number and Depth
   - Block number of the first group record (internal information)
   - Number of entries in the user's group list (internal information)
   - Default logon group
   - Last Logoff Time
   - Last Logon Time
   - Group ID Number with which the user last logged on
   - LU the user logged on to last
   - LU Access Table
   - Capability Level
   - Total CPU Usage for user in all groups
   - Total Connect Time for user in all groups

2. *USER.GROUP Information*: information associated with the user only when logged on associated with the group for which it is defined.

   - Start-up Program to run
   - Logoff Program / Command File
   - Working Directory Name
   - CPU Usage and Connect Time Totals within a specific group
   - CPU Usage and Connect Time Limits within a specific group

The information is divided this way:

- Because a user is a single entity: one real name, one password, one user ID number, and comprehensive CPU usage and connect time totals.
- To enforce the definition that one user has one set of resources: one LU access table and one UDSP number and depth.
- To enforce security: one capability level.
- To enable the system manager to tailor the working environment of the user to the application needed for that user in each group: one start-up program, one logoff program/command file, and one working directory per group.
- For finer control of accounting and accounting limits: CPU usage and connect time totals, and limits for the user in each group.

MASTERACCOUNT File

The MASTERACCOUNT file is a protected system file on the /USERS directory. This file is created by GRUMP during the initialization phase and contains the logon name and corresponding user ID number for all system users. It should never by altered by users.

The first record contains system information with each of the remaining records containing the logon name of a system user. The contents of the record is the user logon name (maximum of 16 characters). The user identification number is the same as the record number (example, user ID 26 is record number 26 in the MASTERACCOUNT file) so it is easy to determine the user name given the user ID. If you have the user ID number and need to know the name of any user, use the subroutine IDToOwner.

Given the user name, this file is not searched to determine the user ID because users may have been deleted or there may have been an unrecoverable error in the creation of a new user. In this case, use subroutine OwnerToID.

Group Configuration File

The operating system maintains information on all group accounts in files called group configuration files. The file resides on the /USERS directory (which should be write protected for security) and its name corresponds to the group's logon name with the type extension .GRP. This distinguishes it from user configuration files on /USERS.

CAUTION

Do not purge a group configuration file from the /USERS directory with the purge command in CI. Use the PURGE GROUP command in GRUMP which purges the group from the multiuser account system with all the necessary cleanup.

The information in the group configuration file is filled in during the creation of a new group account with the GRoup and User Management Program (GRUMP). The system manager or a user with the required capability supplies some of the information and GRUMP generates the rest. For example, GRUMP uses the MASTERGROUP file to assign the group identification number when the new group is created. GRUMP also can be used to list or modify information in an existing group configuration file.

Routines GetAcctInfo, ResetAcctTotals, and SetAcctLimits can be used to programmatically access and alter group information. See the Relocatable Libraries Reference Manual.

Each group configuration file contains the following information:

- Group identification number
- Group totals for CPU usage and connect time
- Group limits for CPU usage and connect time
- Group LU access table
- Number of records in its members list (internal information)
- List of member records

The system uses this file when it creates a user session to check that the user is a member of the group, that the group accounting limits have not been exceeded, and to include the group's resources in the operating environment of the user's session. When a group member logs off, the CPU usage and connect time totals are updated in the group configuration file.

MASTERGROUP File

The MASTERGROUP file is a protected file on the /USERS directory. This file is created by GRUMP during the initialization phase and contains the logon name and corresponding group ID of all the groups on the system.

The first record contains the last group identification number assigned by the system and other system information. The second through the last record contain the logon name of each group defined in the system. A group's identification number is its corresponding record number in the MASTERGROUP file, with the maximum group ID number being 2047. The contents of the record is the group's logon name (maximum of 16 characters). It is easy to determine the group name given the group ID. This file is never searched to determine the group ID given the group name.

If there is a need to know the ID of a group for which the group name is known, use subroutine GroupToID.

If there is a need to know the name of a group for which the group ID is known, use subroutine IdToGroup.

NOGROUP and Default Logon Group

One requirement of the multiuser account system is that all users belong to group NOGROUP. NOGROUP is a group from which users gain no resources. Its purpose is to allow systems to operate without setting up groups. To operate without groups, all that need be done is to make NOGROUP the default logon group for user accounts with the GRoup and Management Program (GRUMP).

However, if the system has been set up with groups and you do not want a particular user logging on associated with group NOGROUP, you can set that user's connect time limit within NOGROUP to zero. If you do not want any users to log on associated with NOGROUP, set the connect time limit for NOGROUP to zero.

All user account definitions contain a list of groups (one or more) to which the user belongs. Any one of the groups may be declared the default logon group by using GRUMP. This allows users to log on without specifying a group account name in the logon entry while automatically associating the session with the default logon group.

Initializing the Multiuser Account System

The Group and User Management Program (GRUMP) creates and manages the multiuser account system. The account system requires a directory called /USERS, which contains a subdirectory HELP and the files MASTERACCOUNT, MASTERGROUP, LOGONPROMPT, MANAGER, NOGROUP.GRP and SYSTEM.GRP. All of these are created automatically by GRUMP if there is no /USERS directory when it is scheduled. GRUMP initializes the multiuser account system in these steps:

1. Prompts for the LU where the directory /USERS is to reside.
2. Creates directories /USERS and /USERS/HELP (for the GRUMP help files).
3. Creates the MASTERACCOUNT file.
4. Reserves record number two and three in the MASTERACCOUNT file for SYSTEM and MANAGER, respectively.
5. Sets the last assigned user identification number to 3 in the MASTERACCOUNT file.
6. Initializes the LOGONPROMPT file, the default is

   ```
   Please log on:
   ```

7. Creates the MASTERGROUP file and sets the last assigned group identification number to 2. Records 3 through 2048 are initialized to zeros.
8. Creates the group configuration file for NOGROUP and sets all the attributes which can never be modified: ID #0, no LUs set in the LU access table, and

make MANAGER a member. The CPU and connect time limits are set to infinity but can be modified.

9. Creates the group configuration file for SYSTEM and sets all the attributes which can never be modified: ID #2, no CPU or connect time limits, make MANAGER a member. All LUs are put in the LU access table, but they can be modified.

10. Creates a user configuration file for MANAGER and sets the following attributes which can never be modified: Capability Level of 31, all LUs put in LU access table, create group entries for NOGROUP and SYSTEM. Then prompts to see if the operator wants to make any modifications to the MANAGER account definition.

Re-initializing Multiuser Account System

WARNING

Read all instructions carefully. Unless done carefully and in correct sequence, re-initializing the multiuser account system can cause serious consequences.

Do NOT log off after you have purged /USERS and before you have re-initialized the multiuser account system or you will not be able to log on again.

If you already have a multiuser account system set up, but want to start over from scratch, follow these steps:

1. Make sure no other users are logged on.

2. Copy all the GRUMP help files from /USERS/HELP to a temporary directory with the purge option.

3. Purge everything in /USERS.

4. Purge /USERS directory.

5. Run GRUMP immediately to re-initialize the multiuser account system.

6. Copy the help files from the temporary directory to /USERS/HELP that GRUMP created during the initialization.

7. Create the new multiuser system.

NOTE

You will have to redefine ownership and associated groups for almost all volumes and directories because they are based on user and group identification numbers which very likely have changed. Groups NOGROUP and SYSTEM are always guaranteed the same group identification numbers. User SYSTEM will always have the same user ID. User MANAGER will always have user ID three when created with GRUMP, but could have a different user ID if it was created prior to Rev. 5000.

GRUMP Utility

The GRoup and User Management Program (GRUMP) is a command-driven program for managing a multiuser account system.

Commands are provided in a command file or interactively from a terminal. The operation mode of GRUMP can alternate between interactive and non-interactive with the TRANSFER command. The TRANSFER command can be issued any time GRUMP is waiting for input.

GRUMP produces two kinds of messages: error messages printed to the terminal (and a log file if one is specified) to say that an error has occurred, and information messages on what is happening which are printed to a specified log file and the terminal if the quiet option is not on.

The RINFO and SINFO utilities which reset and display CPU usage and connect time should be used only in systems that are not using groups. If you use RINFO in a system with groups, it will reset the information for the unique user and USER.NOGROUP which may not be what you want. Use GRUMP to reset and display CPU usage and connect time for systems using groups.

NOTE

If Security/1000 is not installed on your system or is turned off, GRUMP's scheduled name may change temporarily to GRMP0, GRMP1,...or GRMP9 during execution. Be aware that GRUMP tries to maintain the scheduled name as much as possible, but it may vary.

WARNING

Do not remove GRUMP's ID segment (OF GRUMP) because this action can corrupt multiuser files.

Running the GRUMP Utility

The GRUMP runstring is as follows:

```
[RU] GRUMP[ InputSource[ LogFile]][ +Q]
```

InputSource	user's terminal LU or file descriptor for a command file. If not specified or if a user's terminal LU is specified, GRUMP enters interactive mode. If a file descriptor is specified, GRUMP executes the command file.
LogFile	user's terminal LU or file descriptor to which all prompts, responses, error and information messages are to be written. If not specified, all prompts, responses, and messages are printed to your terminal LU.
+Q	quiet option that suppresses information messages.

The parameters are position dependent; therefore, an unspecified parameter must be delimited with commas. For example, to default the InputSource parameter to your terminal LU and specify MYFILE as the log file, you would enter the following command:

```
CI> grump,,myfile
```

The +Q option must be the last parameter in the runstring if an input source and log file are specified. Space holders for input source and log file need not be entered if +Q is the only option used.

Except responses, everything written to the log file is preceded by an asterisk-blank (*) to make the line a comment line. Comment lines will not be executed if the log file is later used as an input source. TR commands are preceded by an asterisk-arrows (* >>) so they become comments indicating the source of subsequent commands.

Running GRUMP Interactively

To run GRUMP interactively, enter the GRUMP runstring without specifying an input source; for example:

```
CI> grump
GRUMP> _
```

In interactive mode, you can enter a GRUMP command after each GRUMP prompt. If the GRUMP command you enter has required parameters, you can specify values for the parameters when entering the command. If you do not specify any or all parameter values, GRUMP will prompt you to enter the unspecified values.

The following example shows the start of creating a new group called MAX:

```
CI> grump
GRUMP> ne g
Enter Group Logon Name: max
.
.
.
```

In the following example, a new user is created with the name "Jamala," real name "James Alabama," password "pup," capability level "12," and UDSP "4:4."

```
CI> grump
GRUMP> ne u jamala `James Alabama`.pup.12./e.4:4

Creating user JAMALA


All users must be in group NOGROUP.


Enter information for JAMALA.NOGROUP


Enter working directory name [::JAMALA]: _
```

If you are entering commands interactively and GRUMP encounters an error (an illegal value in a CPU usage limit, for example) GRUMP prints an error message and again prompts you for the value or information.

Running GRUMP with a Command File

To run GRUMP with a command file, enter the GRUMP runstring and specify a file containing GRUMP commands as the input source; for example:

```
CI> grump newacct
```

Non-interactive commands are supplied by command files and no prompts are issued. Use caution when working with command files because errors within them can cause unexpected results. You should always use a log file in conjunction with your command file to help you check results and trace errors.

If you are using a command file, GRUMP prints errors to the screen and log file and tries to continue. If GRUMP is unable to recover from the error, it reports the error, flushes the current input string, goes interactive, and again prompts for the input. Use the log file to locate the errors in the command file. After correcting the command file, use GRUMP to undo any changes made by the command file before reexecuting the command file.

Appendix F contains an example of a log file used in conjunction with a command file.

GRUMP Command Summary

GRUMP commands fall into five categories. Tables 3-1 through 3-5 group GRUMP commands by function.

Table 3-1. General GRUMP Commands

Command	Purpose
/A	Aborts current command
EX	Terminates GRUMP
HE|?|??	Lists valid commands and help for individual commands
KI	Immediately terminates current user session
RU	Clones and runs a program from GRUMP
TR|/TR	Transfers control from one LU or file to another

Table 3-2. GRUMP Commands for Adding Accounts

Command	Purpose
NE G	Creates a group configuration file for a new group
NE U	Creates a user configuration file for a new user

Table 3-3. GRUMP Commands for Modifying Accounts

Command	Purpose
AL G	Alters attributes defined for groups
AL U	Alters attributes defined for users
PA	Alters user password in the user configuration file
RE G	Resets group accounting information
RE U	Resets user accounting information

Table 3-4. GRUMP Commands for Displaying Account Information

Command	Purpose
LI G	Lists one or more group account entries
LI U	Lists one or more user account entries

Table 3-5. GRUMP Commands for Purging Accounts

Command	Purpose
PU G	Removes a group from the accounting system
PU U	Removes a user from the accounting system

GRUMP Command Conventions

The following conventions apply to all GRUMP commands:

- Only the first or first two characters of the command are checked, all other characters are ignored.
- GRUMP prompts for required command characters and parameters that are not entered at the GRUMP> prompt. For example,

```
GRUMP> ne
Enter (G)roup or (U)sers : _

GRUMP> ne g
Enter group logon name : _

GRUMP> new g general
Enter CPU limit (hh:mm:ss or -1 for No Limit) [No Limit] : _
```

- All GRUMP commands must be entered at the GRUMP prompt. /A, /HE, and /TR can also be entered at the parameter prompts.
- Multiple GRUMP commands cannot be entered in a single input string.
- One command and all its parameters can be entered in a single input string at the GRUMP prompt. The parameters must be entered in the proper order. If an error is encountered, the rest of the string is ignored and GRUMP tries to recover. GRUMP reprompts for any omitted values.
- Enclose character strings with embedded commas or blanks in back quotes (``) or GRUMP reads the blanks and commas as parameter separators. Back quotes can be omitted if the string is the only input on a line or is the last parameter in a multiple value string. Use caution when omitting back quotes. The following examples all give the same result:

1. GRUMP> ne u jamala `James Alabama`.pup.12./e.4:4

2. GRUMP> ne u jamala
   Enter user's real name : James Alabama
   Enter user's password: pup.12./e
   Enter #UDSPs:depth : 4:4

3. GRUMP> new u jamala james alabama
   Enter user's password: pup.12./e.4:4

The result of each example is the creation of user JAMALA with:

real name	James Alabama
password	pup
capability level	12
LU Access Table	all bits set
UDSPs:depth	4:4

Abort (/A)

Purpose: Aborts the current command or subfunction within the ALTER USER command.

Syntax: `/A` or `ctl-D`

Description:

/A is identical to EXIT when entered at GRUMP> prompt (global command).

Note that you must use ctl-D to abort from the prompt "Enter working directory name[xx]:" because /A is considered a valid response. In other words, "/A" is interpreted as a working directory name.

If /A is entered within any command except ALTER USER, the command has no effect on the related multiuser files. See the ALTER USER command for its different use of the /A command.

Alter Group (AL G)

Purpose: Modify the attributes defined for a group.

Syntax: `AL G group`

group — Name of the group account to be modified
Legal values for group are:

groupName	modify the group account specified.
@	modify all existing groups.

Description:

The modifications you make with ALTER GROUP do not affect users currently logged on, only users logging on associated with that group after the alterations have been made to the group configuration file.

GRUMP prompts for values for all the attributes that can be modified with this command. Current values are listed as defaults with [X...X] meaning default.

- Group name
- CPU usage
- Connect time limits
- LU access table

GRUMP returns the message "Capability level not high enough to alter group..." when you do not have the required capability to change an attribute and proceeds to the next attribute.

The prompts for ALTER GROUP are:

```
Enter group logon name:
Enter new group logon name [XXXXX]:
Enter CPU limit (hh:mm:ss or -1 for No Limit) [XX]:
Enter connect time limit (hh:mm:ss or -1 for No Limit) [XX]:
LU access table modifications ( [-]LU#[:LU#2] ):
```

Group Name

This is the group logon name. The name must be unique, follow file naming conventions, and be one to sixteen characters with no parentheses "(" or ")". This definition is not case sensitive so upper or lower case do not matter.

CPU Usage Limit

This defines the CPU limit for the group. The values for CPU limit are:

hh:mm:ss	hh - hours (up to 5960)
or	mm - minutes (up to 59)
hh:mm	ss - seconds (up to 59)
0	inhibits CPU use by group
-1	group has unlimited use of CPU
<cr>	no change to value

Connect Time Limit

This defines the connect time limits for the group. The values for connect time limits are:

hh:mm:ss	hh - hours (up to 596500)
or	mm - minutes (up to 59)
hh:mm	ss - seconds (up to 59)
0	inhibits group from logging on
-1	gives group unlimited connect time
<cr>	no change to value

LU Access Table

The LU access table defines the LUs to which the group has access. The values are:

n	adds LU #n to LU Access Table
-n	removes LU #n from LU Access Table
m:n	adds LU #m to LU #n to LU Access Table
-m:n	removes LU #m to LU #n from LU Access Table
/L	lists LUs set in LU Access Table that group can access
/E	ends LU Access Table modifications
<cr>	ends LU Access Table modifications if only input at prompt

GRUMP continues to prompt for alterations until a /E is encountered in an input string or a <cr> is entered at the LU access table modifications prompt.

Multiple input must be separated by a comma or space as in this example where access to LUs 5, 6, 10 through 20, and 30 is given and access to LUs 40 and 50 through 55 is removed:

```
LU access table modifications ( [ - ]LU#[:LU#2] ): 5,6,10:20,30,-40,-50:55
```

Alter User (AL U)

Purpose: Modify certain attributes of an existing user account.

Syntax: `AL U UserGroup`

UserGroup — User and group information. Can be specified as follows:

user	Information unique to user and certain attributes in the user.NOGROUP record.
user.	Information unique to user regardless of group.
user.group	User.group information about user within specified group.
@.group	Corresponding user.group information of all members of the specified group. GRUMP

	loops through the list of members and modifies member attributes one at a time.
user.@	User.group attributes of user in all groups in which a member. GRUMP loops through user's group list and modifies the attributes in each of the user.group records one at a time.

Description:

You can use GRUMP to modify the attributes for which you have the required capability. If the user does not have the required capability level, GRUMP returns the message "Capability level not high enough to alter user..." and proceeds to the next attribute.

If an attribute (such as the name of the SYSTEM group) can never be modified, GRUMP returns the message "XXXXX attribute can never be modified" and proceeds to the next attribute.

Invalid input is ignored and prompts re-issued. If input is read from a transfer file, GRUMP goes into the interactive mode before re-issuing the prompt.

To add a user to a group or change the default logon group, enter "USER." as the USER.GROUP parameter. GRUMP prompts for modifications to the unique user attributes. Then it prompts for the group name(s) to which the user should be added, for the USER.GROUP information needed for each group, and for the default logon group.

You cannot remove a user from a group with the ALTER USER command. Use the PURGE USER command to do this.

Use of /A in the ALTER USER command differs from use of /A in the other commands because ALTER USER is divided into four steps:

1. Altering unique user information
2. Altering information for a user within a group in which a member
3. Adding a user to a group where not a member and defining the user attributes within that group
4. Changing the default logon group.

If you have completed one or more steps and enter an /A within a subsequent step, modifications made in the step you are in are not acted upon and the ALTER USER command is aborted. Modifications made in any previous step are permanent. Also, you may add the user to several groups in Step 3. Once you have defined all the information for the user within a group, the user will not be removed from that group by a subsequent /A in Step 3 or Step 4.

You cannot modify the capability level, CPU and connect time limits, and LU Access Table for the user MANAGER.

You can use the ALTER USER command to:

- Modify unique user information
  - Logon name
  - Real name
  - Password
  - Capability level
  - LU Access Table
  - Number and depth of UDSPs
- Modify user information within a group
  - Default working directory
  - Startup command
  - Logoff program/command file
  - CPU and connect time limits
- Add a user to a group
- Define default logon group

See the examples at the end of the ALTER USER section for the command prompts.

Logon Name

This is the logon name for the user. The logon name must be unique, follow the file naming conventions, be one to sixteen characters with no parentheses, "(" or ")". This definition is not case sensitive so uppercase or lowercase do not matter.

Real Name

This is the user's real name. The real name may not be more than thirty characters. This definition is case sensitive. The name must be enclosed in back quotes if it is a parameter in a multiple parameter input string.

Password

You can define a password for the user. The password can be no more than fourteen characters with no commas or blanks. Special characters are not recommended. GRUMP issues the prompt:

```
Change password to no password (Yes/No) [N]: _
```

if a password is already defined and you enter a <cr> (indicating no password) at the enter password prompt. This prevents accidentally erasing an existing password. You are not required to know an existing password to change it.

Capability Level

This is the user's capability level. The capability level is an integer value from 0 (lowest) through 31 (highest). Capability level 31 sets the superuser bit in the user configuration file for systems not using Security/1000. A value less than 31 clears the superuser bit in systems not using Security/1000. Any assigned capability level can only be equal or less than your own in both sessions.

LU Access Table

The LU access table defines the LUs which can be accessed. You can remove or add any LUs from the user's LU Access Table. The values are:

n	adds LU #n to LU Access Table
-n	removes LU #n from LU Access Table
m:n	adds LU #m to LU #n to LU Access Table
-m:n	removes LU #m to LU #n from LU Access Table
/L	lists LUs set in LU Access Table that user can access
/E	ends LU Access Table modification
<cr>	ends LU Access Table modifications if sole input at prompt

GRUMP continues to prompt for alterations until a /E is encountered in an input string or a <cr> is entered at the LU access table modifications prompt.

Multiple input must be separated by a comma or space as in this example:

```
LU access table modifications ( [ - ]LU#[:LU#2] ): 5,6,10:20,30,-40,-50:55
```

Number and Depth of UDSPs

This defines the number and depth of the User-definable Directory Search Paths for this user. A <cr> leaves the values as they are. The number and depth of UDSPs must be separated by a colon with no spaces between the numbers and the colon. The legal values are:

m:n	both zero or both not zero (m = number, n = depth)
m:	modify number only
:n	modify depth only
<cr>	use the default (value is unchanged)

GRUMP prints a message indicating that unique user information was modified and terminates unique user information modification. Any changes made up to this point are permanent.

Default Working Directory

This defines the default working directory for the user within the group. A <cr> leaves the value as it is. The directory name must not exceed sixty-three characters including delimiters and must follow the file naming conventions. GRUMP does not verify that the named directory belongs to the user whose account is being modified. The LU containing the working directory must be in the user's LU access table.

If the directory specified does not exist, a GRUMP prompt asks whether the directory should be created.

```
Create directory XXXXX (Yes/No) [N]: _
```

If the directory is to be created, GRUMP prompts for the LU on which it should be created.

```
What LU should the directory go on [0]: _
```

The default LU, 0, is the same as the current working directory or the lowest numbered disc LU on which directories can be created.

Startup Command

This defines the startup command (for the user in the group) which is the program to be executed when the user logs on. The LU containing the directory on which the logoff program/command file must be in the user's LU access table. A <cr> leaves the value as it is. The command can have a maximum of 80 characters.

```
Enter the startup command [RU,CI.RUN::PROGRAMS]: ru ci.run::programs logon.cmd
```

Starting programs other than CI must be loaded and in the directory specified in the command. A startup command string must be enclosed in back quotes if it is a parameter in a multiple parameter string.

```
GRUMP> alter user jamala.lab..`ru ci.run::programs logon.cmd`...100:0:0
```

Logoff Program/Command File

This defines the logoff program/command file (for the user in the group) to be scheduled or to which transferred when the user exits CI. The logoff program must be loaded and on the directory specified. The logoff command file's directory location must be specified. The LU containing the directory on which the logoff program/command file must be in the user's LU access table.

The logoff program/command file must be enclosed in back quotes (``) if it is a parameter in a multiple parameter string. A <cr> at the prompt specifies that the logoff program/command file is undefined. If a logoff program/command file is defined and you enter a <cr>at the 'Enter Logoff program/command file' prompt, GRUMP prompts to ask if the logoff program/command file should be changed to undefined.

```
Enter the logoff program/command file [RU,CLEANUP.RUN::FRED]: <cr>
Change the logoff program/command file to UNDEFINED (Yes/No) [N]: _
```

CPU Usage Limit

This defines the CPU limit for the user within the group. The values for CPU limit are:

hh:mm:ss	hh - hours (up to 5960)
or	mm - minutes (up to 59)
hh:mm	ss - seconds (up to 59)
0	inhibits CPU use by group
-1	group has unlimited use of CPU
<cr>	no change to value

Connect Time Limit

This defines the connect time limits for the user within the group. The values for connect time limits are:

hh:mm:ss	hh - hours (up to 596500)
or	mm - minutes (up to 59)
hh:mm	ss - seconds (up to 59)
0	inhibits group from logging on
-1	gives group unlimited connect time
<cr>	no change to value

Adding a User to a Group

If USER. was entered, GRUMP prompts for unique user information and then if you want to add the user to any existing groups. If you enter YES, GRUMP prompts for a group name and USER.GROUP information for the user in specified group. An /E or <cr> at the "Enter group name" prompt will end prompts for groups. An /A will abort the rest of the ALTER USER command but previous modifications are permanent.

Default Logon Group

This is the name of the user's default logon group. If USER. was entered, GRUMP prompts for the unique user information, existing groups to which the user should be added and then for the default logon group for the user. The user must already be a member of the group that is specified. An /A will abort the rest of the ALTER USER command but previous modifications are permanent.

Alter User Command Examples

If the USER.GROUP parameter is not supplied, the first prompt for all cases will be "Enter user.group parameter:".

* This prompt is dependent upon the response to the prompt immediately preceding it. See the discussion of individual terms for information on the occurrence of asterisked prompts.

1. Altering "USER"

```
GRUMP> al us jamala
Enter new user logon name [JAMALA]:
Enter user's real name [James Alabama]:
Enter password (a <cr> gives no password):
*Change password to no password (Yes/No) [N]
Enter capability level (31=SU) [10]:
LU access table modifications ( [-]LU#[:LU#2] ):
Enter #UDSPs:depth [4:4]:
Enter working directory name [::JAMALA]:
*Create directory ::JAMALA (Yes/No) [N]:
*What LU should the directory go on [0]:
Enter the startup command [RU CI.RUN::PROGRAMS]:
Enter the logoff program/command file [LOGOFF.CMD::JAMALA]:
*Change the logoff program/command file to UNDEFINED (Yes/No) [N]:
Enter CPU limit (hh:mm:ss or -1 for No Limit) [No Limit]:
Enter connect time limit (hh:mm:ss or -1 for No Limit) [No Limit]:
```

2. Altering "USER." (Note the trailing period)

```
GRUMP> al us jamala.
Enter new user logon name [JAMALA]:
Enter user's real name [James Alabama]:
Enter password (a <cr> gives no password):
*Change password to no password (Yes/No) [N]:
Enter capability level (31=SU) [10]:
LU access table modifications ( [-]LU#[:LU#2] ):
Enter #UDSPs:depth [4:4]:

Do you wish to include the user in any existing
group other than NOGROUP (Yes/No) [N]:
*Enter group name (/E or <cr> to end):

Which group should be the default logon group [NOGROUP]:
```

3. Altering "USER.GROUP"

```
GRUMP> al us jamala.nogroup
Enter working directory name [::JAMALA]:
*Create directory ::JAMALA (Yes/No) [N]:
*What LU should the directory go on [0]:
Enter the startup command [RU CI.RUN::PROGRAMS]:
Enter the logoff program/command file [LOGOFF.CMD::JAMALA]:
*Change the logoff program/command file to UNDEFINED (Yes/No) [N]:
Enter CPU limit (hh:mm:ss or -1 for No Limit) [1:00:00]:
Enter connect time limit (hh:mm:ss or -1 for No Limit) [500:00:00]:
```

Exit (EX)

Purpose: Terminates GRUMP.

Syntax: `EX`

`/E`

Help (HE or ?)

Purpose: Displays a summary of GRUMP commands or a brief description of a specific command.

Syntax: `?[?][ command]`

`[/]HE[ command]`

command — Specific GRUMP command to be explained, default is a list of possible GRUMP commands.

Description:

HELP, /HELP, ??, and ? are identical except that /HELP, ??, and ? may be entered from within GRUMP commands while HELP may be entered only at the GRUMP> prompt.

GRUMP will not prompt for the optional command parameter. It lists all the commands. Additional input on the line is read as a command parameter. For example,

Enter password : `/HE /TR cmdfile`

results in a description of the TRANSFER command and sets the password to cmdfile. It will not list all commands and then transfer to cmdfile.

To get help from the "Enter working directory name" prompt, you must use ? rather than /HE because /HE is considered a valid response. In other words, "/HE" is interpreted as a working directory name.

KillSes (KI)

Purpose: Terminates a session.

Syntax: `KI session[ OK]`

session — Session identifier of user to be logged off.

OK — Optional parameter to override the verification prompt.

Description

KILLSES is used to log off a particular user session, remove all programs associated with the session, close and release the associated spool files, and release the user ID entry in the User Table for the session.

Session 0, the system session, cannot be killed.

List Group (LI G)

Purpose: Lists the account information in the configuration file for the group(s) specified

Syntax: `LI G group[ ListFile]`

group	Group name whose configuration file information is to be listed, @ may be specified to indicate all group accounts.
ListFile	Name of a file or terminal LU to which the listing should be sent, default is the terminal.

Example of the LIST GROUP command

```
GRUMP> li g accounting

*************************************************************************

Group:                    ACCOUNTING
Group ID:                 26
Total CPU Time:               1 HR  57 MIN  24 SEC 740 MSEC
Total Connect Time:          44 HR   1 MIN  50 SEC   0 MSEC
CPU Limit:                No Limit
Connect Time Limit:       No Limit

LU Access Table:
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  **  15
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  240 241 242 243 244 245 246 247  **  **  ** 251 252 253 254 255

Members:
     SANDI          SAM          JAMALA          MAX
     HENRY          JUDY

*************************************************************************
```

Note that LUs 14 and 248 through 250 are not in the LU access table.

List User (LI U)

Purpose: Lists the account information in the configuration file for the specified users.

Syntax: `LI U UserGroup[ ListFile]`

UserGroup — User and group information. Can be specified as follows:

user	Unique user information and certain fields in the user.NOGROUP record (for systems not using groups.
user.	Unique user information (regardless of group).
user.group	Unique user information and user.group information about user within the specified group.
@.group	Corresponding user.group information for all members of the specified group. GRUMP lists the group information, then loops through the list of members listing the user.group information for the members one at a time.
user.@	User.group information for the user in all groups in which a member. GRUMP lists the unique user information, then loops through the user's group list listing each of the user.group records one at a time.

ListFile — Name of a file or terminal LU to which the listing should be sent, default is the terminal.

List User Command Examples

1. Example of "USER"

```
GRUMP> li u jamala

*******************************************************************************

User:                          JAMALA
Real Name:                     James Alabama
User ID:                       527
UDSPs:Depth:                   4:4
Capability Level:              10
Default Logon Group:           NOGROUP
Startup Program:               RU CI.RUN::PROGRAMS
Working Directory:             ::JAMALA
Last Logon Time:               Thu Jan 2, 1970 10:00:00 am
Last Logoff Time:              Thu Jan 2, 1970 12:05:31 pm
Group ID Last Logged on:       0
LU Last Logged onto:           101
Total CPU Time:                   1 HR  57 MIN  24 SEC  740 MSEC
Total Connect Time:              44 HR   1 MIN  50 SEC    0 MSEC
CPU Limit:                     No Limit
Connect Time Limit:            No Limit
LU Access Table:
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  **  15
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  240 241 242 243 244 245 246 247  **  **  ** 251 252 253 254 255

*******************************************************************************
```

Note that LUs 14 and 248 through 250 are not in the LU access table

2. Example of "USER."

```
GRUMP> li u jamala.

*******************************************************************************

User:                    JAMALA
Real Name:               James Alabama
User ID:                 527
UDSPs:Depth:             4:4
Capability Level:        10
Default Logon Group:     NOGROUP
Last Logon Time:         Thu Jan 2, 1970 10:00:00 am
Last Logoff Time:        Thu Jan 2, 1970 12:05:31 pm
Total CPU Time:              1 HR  57 MIN  24 SEC  740 MSEC
Total Connect Time:         44 HR   1 MIN  50 SEC    0 MSEC
LU Access Table:
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  **  15
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  240 241 242 243 244 245 246 247  **  **  ** 251 252 253 254 255

Groups:
   NOGROUP         MANUALS            ACCOUNTING

*******************************************************************************
```

Note that LUs 14 and 248 through 250 are not in the LU access table.

3. Example of "USER.GROUP"

```
GRUMP> li u jamala.nogroup

*******************************************************************************

User:                    JAMALA
Real Name:               James Alabama
User ID:                 527
UDSPs:Depth:             4:4
Capability Level:        10
Default Logon Group:     NOGROUP
Last Logon Time:         Thu Jan 2, 1970 10:00:00 am
Last Logoff Time:        Thu Jan 2, 1970 12:05:31 pm
Total CPU Time:               1 HR  57 MIN  24 SEC  740 MSEC
Total Connect Time:          44 HR   1 MIN  50 SEC    0 MSEC
LU Access Table:
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  **  15
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  240 241 242 243 244 245 246 247  **  **  ** 251 252 253 254 255


User.Group:              JAMALA.NOGROUP

Startup Program:         RU CI.RUN::PROGRAMS
Working Directory:       ::JAMALA
Logoff Program/Cmdfile:  Not defined
Total CPU Time:             2 HR  30 MIN  10 SEC  209 MSEC
Total Connect Time:        80 HR   6 MIN  23 SEC   48 MSEC
CPU Limit:               No Limit
Connect Time Limit:      No Limit

*******************************************************************************
```

Note that LUs 14 and 248 through 250 are not in the LU access table.

New Group (NE G)

Purpose: Create a new group by creating a new group configuration file and putting it in the /USERS directory with the type extension .GRP.

Syntax: `NE G`

Description:

You must be a superuser or have the required capability level to create new accounts with the NEW GROUP command.

New accounts may be created during or following system initialization and are usable as soon as they are defined.

The prompts in creating a group are:

```
Enter group logon name: _
Enter CPU limit (hh:mm:ss or -1 for No Limit) [No Limit]: _
Enter connect time limit (hh:mm:ss or -1 for No Limit) [No Limit]: _
LU access table modifications ( [-]LU#[:LU#2] ): _
```

The following defaults are defined for the group attributes set by the NEW GROUP command:

Attribute	Default Value
Group CPU limit	-1 (no limit)
Group conntect time limit	-1 (no limit)
LU access table	all LUs

Group Logon Name

This the the group logon name. The name must be unique, follow the file naming conventions, and be one to sixteen characters with no parentheses, "(" or ")". The definition is not case sensitive so uppercase or lowercase do not matter.

CPU Usage Limit

This defines the CPU limit for the group. The values for CPU limit are:

hh:mm:ss	hh - hours (up to 5960)
or	mm - minutes (up to 59)
hh:mm	ss - seconds (up to 59)
0	inhibits CPU use by group
-1	group has unlimited use of CPU
<cr>	no change to value

No more members may log on after the CPU limit has been exceeded until you rectify the situation by recording the usage and resetting the usage with the RESET GROUP command or by altering the CPU limit for the group. Users logged on when the limit is exceeded are not affected.

Connect Time Limit

This defines the connect time limit for the group. The values for connect time limit are:

hh:mm:ss	hh - hours (up to 596500)
or	mm - minutes (up to 59)
hh:mm	ss - seconds (up to 59)

0	inhibits group from logging on
-1	gives group unlimited connect time
<cr>	no change to value

No new members may log on if the connect time limit has been exceeded until you rectify the situation by recording the connect time and resetting it with the RESET GROUP command or by altering the connect time limit for the group. Users logged on when the connect time limit is exceeded are not affected.

LU Access Table

The LU access table defines the LUs to which the group has access. The values for the LU Access Table are:

n	adds LU #n to LU Access Table
-n	removes LU #n from LU Access Table
m:n	adds LU #m to LU #n to LU Access Table
-m:n	removes LU #m to LU #n from LU Access Table
/L	lists LUs set in LU Access Table that group can access
/E	ends LU Access Table modification
<cr> prompt	ends LU Access Table modifications if sole input at

Multiple input must be separated by a comma or space. Access to all LUs is the default, remove those that you do not want the group to access. Setting an LU bit that has already been set causes no modification and no warning message is sent.

Example of Group Account Creation

Group has unlimited CPU and connect time and has access to LUs 0, 8, and 24.

```
CI> grump
GRUMP> new group
Enter group logon name : sample
Creating group SAMPLE
Enter CPU limit (hh:mm:ss or -1 for No Limit) [No Limit]: <cr>
Enter Connect time limit (hh:mm:ss or -1 for No Limit) [No Limit]: <cr>
LU access table modifications ( [-]LU#[:LU#2] ): -1:23
LU access table modifications ( [-]LU#[:LU#2] ): 8
LU access table modifications ( [-]LU#[:LU#2] ): -25:255
LU access table modifications ( [-]LU#[:LU#2] ): <cr>
GRUMP> _
```

LU Access Table was altered by removing LUs 1 to 23 and 25 to 255 and adding back LU 8. The same modification could have been made with the single string:

```
LU access table modifications ( [-]LU#[:LU#2] ): -1:23 8 -25:255 /e
```

or at the GRUMP> prompt:

```
GRUMP> ne g sample -1 -1 -1:23 8 -25:255 /e
```

or with commas marking default values:

```
GRUMP> new g sample...-1:23 8 -25:255 /e
```

New User (NE U)

Purpose: Creates a new user configuration file on the /USERS directory.

Syntax: `NE U`

Description:

You must be a superuser or have the required capability level to create new accounts with the NE U command.

New accounts may be created during or following system initialization and are usable as soon as they are defined.

The prompts for creating a user are:

```
 Enter user logon name:
 Enter user's real name [???]:
 Enter password (a <cr> gives no password):
 Enter capability level (31=SU) [10]:
 LU access table modifications ( [-]LU#[:LU#2] ):
 #UDSPs:depth [0:0]:
 Enter working directory name [::name]:
*Create directory ::name (Yes/No) [N]:
*What LU should the directory go on [0]:
 Enter the startup command [RU CI.RUN::PROGRAMS]:
 Enter the logoff program/command file [Not Defined]:
 Enter CPU limit (hh:mm:ss or -1 for No Limit) [No Limit]:
 Enter connect time limit (hh:mm:ss or -1 for No Limit) [No Limit]:

 Do you wish to include the user in any existing
 group other than NOGROUP (Yes/No) [N]:
*Enter Group Name (/E or <cr> to end):
*Should this be the default logon group (Yes/No)[N]:
```

* This prompt is dependent upon the response to the prompt immediately preceding it.

The following defaults are defined for the user attributes set by the NEW USER command:

Attribute	Default Value
User's real name	???
User password	no password assigned
Number and depth of UDSPs	0:0 (no UDSPs)

LU access table	all LUs
Default logon group	NOGROUP
Working directory	::LogonName
Startup command	RU CI:PROGRAMS
Logoff program/command file	undefined
CPU limit	-1 (no limit)
Connect time limit	-1 (no limit)

User Logon Name

This is the logon name for the user. The name must be unique, follow the file naming conventions, be one to sixteen characters with no parentheses, "(" or ")". The definition is not case sensitive so uppercase and lowercase do not matter.

User's Real Name

This is the user's real name. The real name may be no more than 30 characters. This definition is case sensitive. The name must be enclosed in backquotes (``) if it is a parameter in a multiple input string. A <cr> gives the default value of ???.

Password

You can define a password for the user. The password can be no more than 14 characters with no commas or blanks. Special characters are not recommended. The default is no password.

Capability Level

This is the user's capability level (USERCPLV). The capability level is an integer value from 0 (lowest) through 31 (highest). A capability level of 31 sets the superuser bit in the user configuration file for systems not using Security/1000. The default capability level is 10 with 0 through 30 as the non-superuser levels in environments without Security/1000. You can assign capability levels which are equal to or lower than your own.

LU Access Table

The LU access table defines the LUs which can be accessed. The LU Access Table values are:

n	adds LU #n to LU Access Table
-n	removes LU #n from LU Access Table
m:n	adds LU #m to LU #n to LU Access Table

-m:n	removes LU #m to LU #n from LU Access Table
/L	lists LUs set in LU Access Table that group can access
/E	ends LU Access Table modification
<cr>	ends LU Access Table modifications if sole input at prompt

Multiple input must be separated by a space or comma. GRUMP will prompt for LU Access Table alterations until a /E is entered in the input string or a <cr> is the only input entered at the prompt. The default is access to all LUs. You remove those you do not wish the user to access.

Be sure the user has access to LU 0, used for program-to-program communication. Also make sure the user can access the LUs containing directories such as /PROGRAMS and the default directory needed.

Number and Depth of UDSPs

This defines the number and depth of the User-definable Directory Search Paths (UDSPs) for this user. The number and depth of UDSPs must be separated by a colon with no spaces between the numbers and the colon. The legal values are:

m:n	- both zero or both not zero (m = number, n = depth)
m:	- modify number only
:n	- modify depth only
<cr>	- use default of 0:0

USER.NOGROUP

The information required after this point defines associated groups for the user. The information you enter applies first to NOGROUP group and then to any other associated groups.

Working Directory

This defines the default working directory for the user when associated with the group. The directory name must not exceed 63 characters including delimiters and must follow directory naming conventions. GRUMP does not verify that the directory belongs to the user whose account is being created.

If the directory specified does not exist, GRUMP prompts to ask whether it is to be created and on which LU it should be created.

```
Create directory XXXXX (Yes/No) [N]: _
What LU should the directory go on [0]: _
```

The default of 0 creates the working directory on the same LU as your current working directory or the lowest numbered disc LU on which directories can be created. The LU on which the working directory resides must be in the user's LU access table. The second prompt is issued only if the directory should be created.

Startup Command

This defines the startup command which is the program to be executed when the user logs on associated with the group. The command can have a maximum of 80 characters. Starting programs other than CI must be loaded and in the directory specified in the command. The LU containing the directory in which the startup command resides must be in the user's LU access table. The startup command string must be enclosed in back quotes (``) if it is a parameter in a multiple parameter input string.

Logoff Program/Command File

This defines the logoff program/command file to be scheduled or to which execution is transferred when the user is associated with the group and exits CI. The logoff program must be loaded and on the directory specified. The directory location of any logoff program/command file should also be specified. The LU containing the directory on which the logoff program/command file resides must be in the user's LU access table. The program or command file must be enclosed in back quotes (``) if it is a parameter in a multiple parameter input string. A <cr> at the prompt gives no logoff program/command file.

CPU Usage Limit

This defines the CPU limit for user when associated with the group. The values for CPU limit are:

hh:mm:ss	hh - hours (up to 5960)
or	mm - minutes (up to 59)
hh:mm	ss - seconds (up to 59)
0	inhibits CPU use by group
-1	group has unlimited use of CPU
<cr>	no change to value

No new members may log on after the CPU limit has been exceeded until the you rectify the situation by recording the CPU usage and resetting it with the RESET USER command or by altering the CPU limit for the group. Users logged on when the limit is exceeded are not affected.

Connect Time Limit

This defines the connect time limit for user when associated with the group. The values for connect time limit are:

hh:mm:ss	hh - hours (up to 596500)
or	mm - minutes (up to 59)
hh:mm	ss - seconds (up to 59)
0	inhibits group from logging on
-1	gives group unlimited connect time
<cr>	no change to value

No new members may log on to the account if the connect time limit has been exceeded until you rectify the situation by recording the connect time and resetting it with the RESET USER command or by altering the connect time limit for the group. Users logged on when the connect time limit is exceeded are not affected.

Associated Groups

GRUMP prompts to see if the user should be included in groups other than NOGROUP. If other groups are specified, GRUMP prompts for the same information required for USER.NOGROUP (working directory, startup command, logoff program, CPU and connect time limits). GRUMP prompts for additional group entries and information until a /E or <cr> is entered.

Password (PA)

Purpose: Changes a current user account password.

Syntax: `PA[ username]`

username	User account name which will have its password changed, default is the user's account.

Description:

You must know the present password to change it with this command

Password modification steps:

```
Present password:
New password:
Retype new password:
```

The password input is not echoed to the terminal.

After verification of new password, GRUMP changes the password in the configuration file.

Purge Group (PU G)

Purpose: Removes a group from the multiuser account system. The group configuration file is purged from the /USERS directory. The group record and information are removed from the configuration file of all group members

CAUTION

Do not purge group configuration files from the /USERS directory with the PU command in CI. The PURGE GROUP command in GRUMP purges the group from the multiuser account system with all the necessary cleanup.

Syntax: `PU G groupname[ OK]`

groupname	Group whose account information is to be purged."@" will purge all groups with the exception of the NOGROUP and SYSTEM groups.
OK	Optional parameter to override verification prompt.

Description:

A group account cannot be purged if any user associated with the group is logged on.

GRUMP issues a verification prompt if optional OK parameter not used.

Purge User (PU U)

Purpose: Removes a user from the multiuser account system. The user configuration file is removed from the /USERS directory. The user record is removed from the members list in all groups in which a member.

CAUTION

Do not use the PU command in CI to purge user configuration files from the /USERS directory. The PURGE USER command in GRUMP purges the user from the multiuser account system with all the necessary cleanup.

Syntax: `PU U UserGroup[ OK]`

UserGroup	User and group information. Can be specified as follows:	
	user[.]	User name to be removed from all groups in which a member and user account to be purged.
	user.group	User name to be removed from specified group.

@.group	All members of a specified group to be removed, group account not changed, not allowed for NOGROUP group, MANAGER cannot be removed from SYSTEM group.
user.@	User to be removed from all groups with the exception of NOGROUP, user account not changed, MANAGER cannot be removed from SYSTEM.

OK — Option parameter to override the verification prompt.

Description:

A user account cannot be purged if anyone is logged on (associated with any group) as that user.

USER.GROUP account information cannot be purged if the user is logged on associated with the specified group.

GRUMP issues a verification prompt if OK parameter not used.

Reset Group (RE G)

Purpose: Clears the CPU usage and/or connect time total(s) for all groups or a specific group.

Syntax: `RE G group[ CP|CO]`

group	One or more groups whose totals will be reset, "@" means all group totals are to be reset.
CP	Optional parameter that resets the CPU usage for the specified group.
CO	Optional parameter that resets the connect time total for the specified group.

Description:

Resetting a group's totals does not affect the individual USER.GROUP totals for group members. User totals can be reset with the RESET USER command.

If neither the CP or CO parameter is specified, both totals are reset. CP and CO cannot be specified at the same time.

Reset User (RE U)

Purpose: Clears the CPU usage and connect time totals for a specific user (grand totals), a single user within a group (USER.GROUP totals), all users within a group (USER.GROUP totals), or a user within all groups where a member (USER.GROUP totals).

Syntax: `RE U UserGroup[ CP|CO]`

UserGroup	User and group information. Can be specified as follows:	
	user	Account totals unique to the user and appropriate totals in user.NOGROUP record for systems not using groups. User.group totals are not affected.
	user.	Cumulative totals for user (sum of all user.group totals).
	user.group	User.group totals for user within specified group.
	@.group	User.group totals of all the members of the group when associated with the specified group at logon. GRUMP loops through the member lists and resets member totals one at a time. The cumulative totals for the user or group are not affected.
	user.@	User.group totals for the user in each group where a member. GRUMP loops through the group list and resets total in the user.group records one at a time. The cumulative totals for the user are not affected.
CP	Optional parameter to reset only CPU total for the specified user.	
CO	Optional parameter to reset only the CONNECT total for the specified use.	

Description:

If neither the CP or CO parameter is specified, both are reset.

Run (RU)

Purpose: Clones and runs a program.

Syntax: `RU program[ ParmString]`

program	Name of the program to run.
ParmString	Parameters for the listed program.

Transfer (TR)

Purpose: Allows the GRUMP user to alternate between interactive and non-interactive command input modes.

Syntax: `[/]TR InputSource|-n`

InputSource	Terminal LU or file name from which commands are read.
-n	Negative integer that causes the control input to go back N levels.

Description:

TR and /TR are identical except that /TR can be issued from within GRUMP commands.

Transfer commands can be nested to a depth of 5 levels. For example, a TR command issued at a terminal transfers control to a command file which contains a command transferring control to a terminal LU. At this point, transfers are nested 2 deep.

The negative integer (-n) transfers control back n levels. If n is greater than the number of nested levels, all previously nested input sources are closed and GRUMP goes into interactive command input mode at the user's terminal.

An end-of-file condition is interpreted as a /TR -1 command and transfers control to previous level.

If an irrecoverable error is encountered in a command file, GRUMP transfers control to the terminal, flushes the input string, and reprompts the user. A subsequent /TR -1 transfers control back to the command file where the error occurred.

Hitting the BREAK key forces transfer to the terminal. A /TR -1 transfers control back to command file in effect at break.

TRANSFER commands are written to the log file:

```
* GRUMP>
* >> TR
* Enter transfer file of -(# of levels back) :
* >> filename/LU
* GRUMP>
```

The first column asterisk makes the commands comments that will not be executed if GRUMP is re-run using the log file as an input source.

The ">>"s highlight the transfer command and the source of subsequent commands.

A command file exited with an implied return, EOF, will appear as:

```
* GRUMP>
* GRUMP>
```

A transfer command used to exit a command file appears the same as shown in the preceding log file example.

A

Security/1000 Error Codes

-1700 Stgen's Symbol Table overflowed.

This is an internal error; contact Hewlett-Packard.

-1701 Security/1000 not generated into your system.

The OS modules SECOS.REL and CHECK.REL must be generated into your system in order to use Security/1000.

-1702 No such Category or Function.

The Category or Function specified cannot be found in the Security/1000 tables. Check your listing of the installed tables. (See LT command in the SECTL utility.)

-1703 Security/1000 NOT turned on.

Certain Security/1000 functions can only be performed if Security/1000 is turned on. See the IN and ON commands in the SECTL utility.

-1704 Security/1000 has not been initialized.

Security/1000 must be initialized in order to turn on or use Security/1000. See the IN command in the SECTL utility.

-1705 Illegal value for CPLV, must be in the range 0-31.

If you are using the Security/1000 tables to store your application's configuration information requiring values outside the 0-31 range, use the routines SecPutCitNam, SecPutCftNam, SecPutCitNum, and SecPutCftNum found in the SEC1000.LIB library.

-1706 Duplicate Category name.

The category name already exists in the Security/1000 tables.

-1707 Duplicate Function name.

The function name already exists within the specified category.

-1708 Specified program NOT found.

An ID segment belonging to the specified program cannot be found.

-1709 Specified file not a SNAP file or corrupt SNAP file.

The supplied file name was not a snap file. If it was a snap file, it was corrupt.

-1710 STGEN found errors.

The table generator, STGEN, found errors. See the list file produced by STGEN for the details.

-1713 Security Violation.

The requested action cannot be allowed because you do not have the required capability level.

B

SECURITY.TBL

SECURITY.TBL is the security source code shipped with the software. It is used with the SECTL GT command to generate a set of tables at installation.

Security Table Source Format

The general format is free form, with exceptions only where noted. Input is NOT case sensitive, it is all upshifted internally.

The NAME fields are all up to six bytes long. They may consist of any ASCII characters except <cr> and <lf>.

Any line with an asterisk (*) in column one is considered a comment. Comments may appear at the end of any source line, provided that ALL fields of the source line were specified.

Categories

Categories are defined with the $CATEGORY keyword. The format is:

```
$category: <category name> [rangeoff]
```

$category can start anywhere on the line. There must be at least one space between the colon (:) and the start of the category name. The rangeoff option instructs the table generator to allow values outside the 0 to 31 range. This option is used when the functions in the category are used to hold information other than CPLVs. Rangeoff cannot be used on special categories.

Special Categories

There are four special categories, each indicated by a key word. Note that the colon (:) is part of the key word. The rangeoff option cannot be specified with these categories

```
$system:    OS commands and signal security
$exec:      Exec call security
$fmp:       File system security
$security:  Security/1000 internal security checking
```

Special categories can appear in any order in the source file and the functions within the category can appear in any order. However, the table generator will reorder the special categories and the functions within them to an order which is known to the system software (for performance reasons).

Only certain functions (predefined) can appear within a special category. These are known to the table generator. If any other function appears in a special category it will generate an error condition. The predefined functions for each special category are listed below.

Some of the special categories have reserved functions in them. Currently no software makes use of them. They are for possible future expansion of the functions in the special categories. The reserved functions will appear in the list file that the SECTL LT command or the SecLisTable subroutine produces.

`$system:` Each function corresponds to an OS command except sglkil which is used by signals to determine who can send a kill signal

```
br, cd, dt, go, of, pr, ss, sz, ul,
up, vs, ws, as, dn, ds, ps, tm, ru,
xq, ex, sigkil
```

`$exec:` Each function corresponds to an exec code. Note that even if an exec code is currently undefined, it is still given an entry within this category. Note also that $exec is currently an inactive category which means that it is not presently used by HP software.

```
exec01, exec02, exec03, ......
......, exec44
```

`$fmp:` Each function corresponds to an FMP function. See the following SECURITY.TBL section for the definitions of the FMP functions.

```
fmp00, fmp01, fmp02, ......
....., fmp35
```

$security: As the functions in this category represent a variety of things, each function and its meaning is listed:

Function	Meaning
edtfnc	subroutine SecEditFunction
rncat	subroutine SecRenameCat
rnfnc	subroutine SecRenameFnc
putprg	subroutine SecPutProgCplv
putrq	subroutine SecPutRqusCplv
putcit	subroutines SecPutCitNam, SecPutCitNum
putcft	subroutines SecPutCftNam, SecPutCftNum

ckcplv	subroutines	SecChkCplvNam, SecChkCplvNum
inital	subroutine	SecInitialize
switch	subroutine	SecSwitch
vfnam	subroutine	VfNam
clgon	subroutine	Clgon
clgof	subroutine	Clgof

Functions

Each category contains a series of functions. There must be at least one function per category. The functions follow the category definition. All functions will be linked to the last seen category definition. The syntax of the function statement is

fncname <base> sub1 sub2 sub3

fncname	the name of the function, must be unique within the current category.
base	The cplv of the base function, must be in the range 0-31, unless rangeoff was specified.
sub1	The cplv of subfunction1, must be in the range 0-31, unless range off was specified.
sub2	The cplv of subfunction2, must be in the range 0-31, unless range off was specified.
sub3	The cplv of subfunction3, must be in the range 0-31, unless range off was specified.

An example,

```
putcit   12   1   2   0
```

SECURITY.TBL

This is a sample of the security source code shipped with the software. Use it to generate a set of tables at installation.

```
*
*-----< Security definition for programmatic access to the file
*-----< system via the FMP library.
*-----< The following meanings have been applied to the base function
*-----< and or subfunctions.
*-----<
*-----<     (1)     Base - caller can either call the routine or not.
*-----<             Sub1 - action can be performed in the current WD.
*-----<             Sub2 - action can be performed on the same LU as
*-----<                    the current WD.
*-----<             Sub3 - action can be performed on any LU.
*-----<
*-----<     (2)     Base - caller can either call the routine or not.
*-----<             Sub1 - action can be performed by the owner.
*-----<             Sub2 - action can be performed by another member of
*-----<                    the owner's group.
*-----<             Sub3 - action can be performed by any other user on
*-----<                    the system.
*-----<
*-----<     (3)     Base - caller can either call the routine or not.
*-----<             Sub1 - not defined
*-----<             Sub2 - not defined
*-----<             Sub3 - not defined
*
```

```
$fmp:

        create   10 10 11 13    * FmpOpen - create mode          (1)
        open     1  1  2  4     * FmpOpen - open mode            (1)
        fmp02    0  0  0  0     * reserved for future use
        purge    10 10 11 13    * FmpPurge                       (1)
        unprg    10 10 11 13    * FmpUnPurge                     (1)
        init     30 0  0  0     * FmpMount - initialize mode     (2)
        mount    5  0  0  0     * FmpMount - mount only mode     (2)
        dismnt   5  0  0  0     * FmpDismount                    (2)
        crdir    10 10 11 13    * FmpCreateDir                   (1)
        wd       5  0  0  0     * FmpWorkingDir                  (3)
        acctim   5  0  0  0     * FmpAccessTime                  (3)
        updtim   5  0  0  0     * FmpUpdatetime                  (3)
        cretim   5  0  0  0     * FmpCreatetime                  (3)
        setown   10 0  0  0     * FmpSetOwner                    (3)
        eof      5  0  0  0     * FmpEof                         (3)
        size     5  0  0  0     * FmpSize                        (3)
        setwd    15 15 16 18    * FmpSetWorkingDir               (1)
        info     5  0  0  0     * FmpInfo                        (3)
        setdir   10 0  0  0     * FmpSetDirInfo                  (3)
        reccnt   5  0  0  0     * FmpRecordCount                 (3)
        filenm   5  0  0  0     * FmpFileName                    (3)
        fmp21    0  0  0  0     * reserved for future use
        rename   10 10 11 13    * Fmprename                      (1)
```

```
            fmp23    0  0  0  0     * reserved for future use
            trunct   10 0  0  0     * FmpTruncate                    (3)
            prot     5  0  0  0     * FmpProtection                  (3)
            setprt   10 0  0  0     * FmpSetProtection               (3)
            opfile   5  0  0  0     * FmpOpenFiles                  (3)
            reclen   5  0  0  0     * FmpRecordLen                  (3)
            fmp29    0  0  0  0     * reserved for future use
            fmp30    0  0  0  0     * reserved for future use
            fmp31    0  0  0  0     * reserved for future use
            dirnam   5  0  0  0     * FmpDirAddToNam                (3)
            fmp33    0  0  0  0     * reserved for future use
            fmp34    0  0  0  0     * reserved for future use
            fmp35    0  0  0  0     * reserved for future use
            fmp36    0  0  0  0     * reserved for future use
            fmp37    0  0  0  0     * reserved for future use
            fmp38    0  0  0  0     * reserved for future use
            fmp39    0  0  0  0     * reserved for future use
            fmp40    0  0  0  0     * reserved for future use
            fmp41    0  0  0  0     * reserved for future use
            fmp42    0  0  0  0     * reserved for future use
            fmp43    0  0  0  0     * reserved for future use
            fmp44    0  0  0  0     * reserved for future use
            fmp45    0  0  0  0     * reserved for future use
            fmp46    0  0  0  0     * reserved for future use
            fmp47    0  0  0  0     * reserved for future use
            fmp48    0  0  0  0     * reserved for future use
            fmp49    0  0  0  0     * reserved for future use
            fmp50    0  0  0  0     * reserved for future use
            fmp51    0  0  0  0     * reserved for future use
            fmp52    0  0  0  0     * reserved for future use
            fmp53    0  0  0  0     * reserved for future use
            fmp54    0  0  0  0     * reserved for future use
            fmp55    0  0  0  0     * reserved for future use
            fmp56    0  0  0  0     * reserved for future use
            fmp57    0  0  0  0     * reserved for future use
            fmp58    0  0  0  0     * reserved for future use
            fmp59    0  0  0  0     * reserved for future use
            fmp60    0  0  0  0     * reserved for future use
            fmp61    0  0  0  0     * reserved for future use
            fmp62    0  0  0  0     * reserved for future use
            dsrtr    15 0  0  0     * ds/1000 file transparency      (3)

*
*------< The System commands security category.  These are the commands
*------< implemented in the kernel.  The program that issues the OS
*------< command is called the "sender" and the program that the
*------< command will affect is called the "receiver".
*------<
*------< The following meanings have been applied to the base function
*------< and or subfunctions.  The sender's PROGCPLV is always checked
*------< against the Base.  If that check is passed, the sender's PROGCPLV
*------< is checked against one of the subfunctions (which subfunction
*------< is used depends on the parameters).
*------<
*------<   Base  - The sender can either issue the command or not.
*------<
*------<   Sub01 - The sender and the receiver are in the same session.
*------<
*------<   Sub02 - The sender and the receiver are in different sessions but
```

```
*-----<           they are both in the same group.
*-----<
*-----<   Sub03 - The sender and the receiver are in different sessions and
*-----<           different groups.
*


$system:
          as       15  15  20  25
          br       5   5   20  25
          cd       6   6   11  16
          dt       10  10  20  25
          go       5   5   10  15
          of       10  10  20  25
          pr       20  20  20  25
          ss       10  10  20  25
          sz       6   6   11  16
          ul       20  0   0   0
          up       5   0   0   0
          vs       7   7   20  25
          ws       12  12  20  25
          dn       15  0   0   0
          ds       5   0   0   0
          ps       0   0   0   0
          tm       1   31  0   0
          ru       8   8   0   0 * sub02 and sub03 undefined for this cmd.
          xq       8   8   0   0 * sub02 and sub03 undefined for this cmd.
          ex       0   0   0   0
          sglkil   10  10  20  20 * Signal kill function

*
*-----< The first of two categories for SECURITY/1000
*-----< This category defines who can use the SECURITY/1000 routines and
*-----< what they can do with them.
*

$security:

              edtfnc  31  0   0   0   * SecEditFunction
              rncat   31  0   0   0   * SecRenameCat
              rnfnc   31  0   0   0   * SecRenameFnc


*
*-----< SecPutProgCplv and SecPutRqusCplv have the base function and
*-----< the subfunction defined.  Their meanings are as follows.
*-----<
*-----<   Base    - The request applies to the calling program.
*-----<
*-----<   Sub01   - The request applies to another program in the
*-----<             same session as the caller.
*-----<
*-----<   Sub02   - The request applies to a program in another
*-----<             session but the same group as the caller.
*-----<
*-----<   Sub03   - The request applies to a program in another
*-----<             session and group from the caller.
*
```

```
            putprg  5   15  31  0   * SecPutProgCplv
            putrq   5   15  31  0   * SecPutRqusCplv
            putcit  31  0   0   0   * SecPutCitNam and SecPutCitNam
            putcft  31  0   0   0   * SecPutCftNam and SecPutCftNam
            ckcplv  0   0   0   0   * SecChkCplvNam and SecChkCplvNum
            inital  31  0   0   0   * Secinitial
            switch  31  0   0   0   * SecSwitch


*
*------<   VfNam, Clgon, Clgof.
*------<
*------<   Base    - Yes/No
*------<
*------<   Sub01   - Log directive.  If non-zero, then any security check
*------<             that fails will be logged to the spool log file.
*------<
*------<   Sub02   - Abort Directive.  If 0, nothing will be aborted.
*------<             If 1, the program that failed the security check
*------<             will be aborted.
*


            vfnam   20  1   1   0       * VfNam
            clgon   20  1   1   0       * Clgon
            clgof   20  1   1   0       * Clgof


*
*------< The second SECURITY/1000 category.  It is used to determine
*------< who can use the Sectl commands and what they can do with them.
*

$category: sectl

            in      31  0   0   0   * IN command
            sw      31  0   0   0   * SW command
            ec      31  0   0   0   * EC command
            he      0   0   0   0   * HE command
            gt      5   0   0   0   * GT command
            lt      5   0   0   0   * LT command
            pc      31  0   0   0   * PC command
            rn      31  0   0   0   * Rn command
            rq      31  0   0   0   * RQ command
            res000  0   0   0   0   * Spares for future features
            res001  0   0   0   0   * Spares for future features
            res002  0   0   0   0   * Spares for future features
            res003  0   0   0   0   * Spares for future features
            res004  0   0   0   0   * Spares for future features
            res005  0   0   0   0   * Spares for future features
            res006  0   0   0   0   * Spares for future features
            res007  0   0   0   0   * Spares for future features
            res008  0   0   0   0   * Spares for future features
            res009  0   0   0   0   * Spares for future features


*
*------< Link security table
*
```

```
$category: link

              os      5   0   0   0   * operator suspend link

*
*-----<  PR command functions have the following meanings.
*-----<
*-----<  Base    -  The user can either issue this command or not.  If so,
*-----<             then the user is subject to the subfunction meanings.
*-----<
*-----<  sub01   -  The user can only decrease the default priority
*-----<             ( 99 -> 32k ).
*-----<
*-----<  sub02   -  The user can increase the priority to 51
*-----<
*-----<  sub03   -  The user can set any priority.
*

              pr      15  15  20  28  * set priority of program

              lc      20  0   0   0   * labelled system common
              sc      20  0   0   0   * blank system common
              sh      20  0   0   0   * Use Sharable EMA

*
*-----< CI security tables.  All CI commands use only the base function.
*-----< Therefore, it is a simple YES/NO decision for each command.
*-----<
*-----< Interdependencies.  A number of CI commands map directly onto FMP
*-----< routines or the OS kernel.  For example, PU ( purge file ) maps
*-----< onto FmpPurge and BR maps directly onto the OS kernel command BR.
*-----< It is therefore possible to pass the CI command security check
*-----< but fail at a deeper level ( FMP or the kernel ).  For a list of
*-----< the interdependencies see the SECURITY/1000 Reference Manual.
*-----<
*-----< CI commands fall into one of the five following
*-----< interdependencies.
*-----<
*-----< (1)   The command is really a program.  There could be up to 3
*-----<       levels of security that have to be passed in order for the
*-----<       command to be allowed.  The 3 levels are:
*-----<
*-----<           - CI security check (can the user issue the
*-----<             command from CI).
*-----<
*-----<           - Program security (does the user have enough
*-----<             capability to run the program that implements
*-----<             the command).
*-----<
*-----<           - FMP or OS kernel command (does the user have
*-----<             enough capability to call the FMP routine or
*-----<             issue the kernel command that the CI
*-----<             command maps onto).
*-----<
*-----< (2)   The CI command maps onto an FMP routine.  Therefore, there
*-----<       are two levels of security involved.
*-----<
*-----<           - CI security check (can the user issue the
```

```
*------<              command from CI).
*------<
*------<            - FMP routines (does the user have the capability
*------<              to call the FMP routine that CI is about to
*------<              use).
*------<
*------< (3)    The CI command maps onto an OS kernel command.  Therefore,
*------<        there are two levels of security involved.
*------<
*------<            - CI security check (can the user issue the
*------<              command from CI).
*------<
*------<            - OS kernel command (does the user have the
*------<              capability to use the OS kernel command that CI
*------<              is about to use).
*------<
*------< (4)    The CI command is handled internally.  In this case there
*------<        is only one level of security involved, CI's.
*------<
*------< (5)    The CI command calls a documented utility routine (as
*------<        found in the Relocatable Library Reference Manual or the
*------<        Programmer's Reference Manual).  To find out which
*------<        routines have security features in them, see the
*------<        $security: category above.
*------<

$category: ci

              as      0   0   0   0   * assign partition          (3)
              at      10  0   0   0   * time schedule            (4)
              br      0   0   0   0   * Set break flag           (3)
              cd      0   0   0   0   * code partition size      (3)
              cl      0   0   0   0   * list mounted dics       (4)
              cn      10  0   0   0   * device control           (4)
              co      0   0   0   0   * copy files              (2)
              cr      0   0   0   0   * create file             (2)
              crdir   0   0   0   0   * create directory        (2)
              dc      0   0   0   0   * dismount dics           (2)
              dl      0   0   0   0   * directory list          (1)
              dt      0   0   0   0   * data partition size     (3)
              echo    0   0   0   0   * echo parameters        (4)
              ex      0   0   0   0   * terminate CI            (4)
              go      0   0   0   0   * resume suspended program  (3)
              ifthen  0   0   0   0   * control structure      (4)
              in      20  0   0   0   * initialize disc        (2)
              io      0   0   0   0   * display I/O config     (1)
              is      0   0   0   0   * control compare        (1)
              li      0   0   0   0   * list files             (4)
              mc      0   0   0   0   * mount disc             (2)
              mo      0   0   0   0   * move files             (4)
              of      0   0   0   0   * stop prog              (3)
              owner   0   0   0   0   * file ownership         (2)
              path    0   0   0   0   * UDSP control          (1)
              pr      0   0   0   0   * program priority       (3)
              prot    0   0   0   0   * file protection        (2)
              ps      0   0   0   0   * program status        (3)
              pu      0   0   0   0   * purge files           (2)
              return  0   0   0   0   * control return        (4)
              rn      0   0   0   0   * rename files           (2)
```

```
            rp      10  0   0   0   * create id segment          (2)
            rs      0   0   0   0   * restart CI                 (1)
            ru      0   0   0   0   * run prog                   (2)
            set     8   0   0   0   * set user CI variable       (4)
            sp      0   0   0   0   * spool operation           (1)
            ss      0   0   0   0   * suspend program           (3)
            sz      0   0   0   0   * modify program size       (3)
            tm      0   0   0   0   * system time               (3)
            to      12  0   0   0   * device time out           (4)
            tr      0   0   0   0   * transfer to cmd file      (4)
            ul      0   0   0   0   * unlock shareable EMA part. (3)
            unpu    0   0   0   0   * unpurge file             (2)
            unset   8   0   0   0   * clear user CI variable   (4)
            up      0   0   0   0   * up a device              (3)
            vs      0   0   0   0   * ema/vma size             (3)
            wd      0   0   0   0   * working directory        (2)
            wh      0   0   0   0   * system status           (1)
            while   0   0   0   0   * control structure       (4)
            whosd   0   0   0   0   * volume users           (1)
            ws      0   0   0   0   * vma working set         (3)
            xq      0   0   0   0   * nowait prog run         (2)
            ?       0   0   0   0   * help                    (4)

*
*------< The security category for GRUMP.
*
*
*------< Meanings of the base and subfunctions are:
*------<
*------< (1) Base    - Capability needed to perform the function on
*------<               yourself.
*------<     Sub01   - Capability needed to perform the function on
*------<               someone else.
*------<     Sub02   - Not defined
*------<     Sub03   - Not defined
*------<
*------< (2) Base    - Capability needed to perform the function on
*------<               group in which you are a member.
*------<     Sub01   - Capability needed to perform the function on
*------<               group in which you are not a member.
*------<     Sub02   - Not defined
*------<     Sub03   - Not defined
*

$category: grump

          algrln  31  31  0   0   *alter group logon name          (2)
          algrcp  25  31  0   0   *alter group cpu limit           (2)
          algrco  25  31  0   0   *alter group connect time limit  (2)
          algrbm  25  31  0   0   *alter group bit map            (2)
          alusln  31  31  0   0   *alter logon user name          (1)
          alusrn  10  31  0   0   *alter logon user real name     (1)
          aluspw  10  31  0   0   *alter user password           (1)
          alusud  25  31  0   0   *alter user udsp depth and levels (1)
          aluscl  31  31  0   0   *alter user cplv               (1)
          alusbm  25  31  0   0   *alter user bit map            (1)
          aluscp  31  31  0   0   *alter user cpu limit          (1)
          alusco  31  31  0   0   *alter user connect limit      (1)
          alusgr  25  31  0   0   *alter user group list (ie.  add   (1)
```

```
                                        *  groups)
          aludlg  15  31  0   0   *alter user default logon group   (1)
          alussu  15  31  0   0   *alter user startup program       (1)
          aluslf  15  31  0   0   *alter user logoff program        (1)
          aluswd  31  31  0   0   *alter user logon working         (1)
                                        *  directory
          ligrp   15  31  0   0   *list group definitions           (2)
          liusr   15  31  0   0   *list user definitions            (1)
          passwd   0  25  0   0   *change user password             (1)

*
*-----< The meanings of the base and subfunctions are now,
*-----<
*-----<   Base    - Capability needed to issue the command.
*-----<   Sub01   - Not defined.
*-----<   Sub02   - Not defined.
*-----<   Sub03   - Not defined.
*

          negrp   31  0   0   0   *create new group
          neusr   31  0   0   0   *create new user
          pugrp   31  0   0   0   *purge a group
          puusr   31  0   0   0   *purge a user
          regrp   31  0   0   0   *reset a groups accounting info
          reusr   31  0   0   0   *reset a users accounting info
          kilses  31  0   0   0   *kill a session

*
*-----< The utils category.
*

$category: utils

      kilses  31  0   0   0   *Kill a session, yes or on

*
*-----< Meanings of functions for seslu.
*-----<
*-----<   Base    - List your sessions's bit map
*-----<   Sub01   - List another session's bit map
*-----<   Sub02   - Modify your session's bit map
*-----<   Sub03   - Modify another session's bit map
*

      seslu   5   10  15  31  *SESLU utility


*
*-----< The following two categories (HP000 and HP001) are reserved
*-----< for use by HP.
*

$category: hp000
      res00  0  0  0  0
      res01  0  0  0  0
      res02  0  0  0  0
      res03  0  0  0  0
      res04  0  0  0  0
      res05  0  0  0  0
```

```
      res06  0  0  0  0
      res07  0  0  0  0
      res08  0  0  0  0
      res09  0  0  0  0
      res10  0  0  0  0
      res11  0  0  0  0
      res12  0  0  0  0
      res13  0  0  0  0
      res14  0  0  0  0
      res15  0  0  0  0
      res16  0  0  0  0
      res17  0  0  0  0
      res18  0  0  0  0
      res19  0  0  0  0

$category: hp001

      res00  0  0  0  0
      res01  0  0  0  0
      res02  0  0  0  0
      res03  0  0  0  0
      res04  0  0  0  0
      res05  0  0  0  0
      res06  0  0  0  0
      res07  0  0  0  0
      res08  0  0  0  0
      res09  0  0  0  0
      res10  0  0  0  0
      res11  0  0  0  0
      res12  0  0  0  0
      res13  0  0  0  0
      res14  0  0  0  0
      res15  0  0  0  0
      res16  0  0  0  0
      res17  0  0  0  0
      res18  0  0  0  0
      res19  0  0  0  0
```

Example of a Setup Program

```
program setup
implicit none

*
*-----< This program renames or edits entries in the security tables.
*-----< It receives command input from a file whose name is supplied
*-----< in the run string.
*-----<
*-----< An example of the command input to this program is given below.
*-----<
*-----<    rn,c,hp000,crdgp
*-----<    rn,f,crdgp,res00,nogroup
*-----<    rn,f,crdgp,res01,system
*-----<    ec,crdgp,system,0,1,2,3
*-----<    ex
*-----<
*-----< The command input format is the same as that accepted by the
*-----< SECTL utility.
*-----<
*-----< This program will work only if it has a high enough capability
*-----< level. The SECURITY/1000 routines will check the capability
*-----< level of this program to determine whether the requested action
*-----< can be allowed.
*

      integer indcb(144),runarray(40),xlog
      integer FmpOpen,error,DecimalToInt,length
      integer inarray(40),j
      integer CatArray(3),OldCatArray(3),NewCatArray(3)
      integer OldFncArray(3),NewFncArray(3),FncArray(3)
      integer CplvArray(4),fmpread

      character runstring*80,instring*80,command*6,cplvvalue*6
      character CategoryName*6,OldCategoryName*6,NewCategoryName*6
      character OldFunctionName*6,NewFunctionName*6,FunctionName*6

      equivalence (runstring,runarray)
      equivalence (instring,inarray)

      equivalence (OldFunctionName,  OldFncArray),
     +            (NewFunctionName,  NewFncArray),
     +            (OldCategoryName,  OldCatArray),
     +            (NewCategoryName,  NewCatArray),
     +            (CategoryName,     CatArray),
     +            (FunctionName,     FncArray)


      instring=' '


*
*-----< Pick up the run string to find the name of the input file.
*

      call getst(runarray,-80,xlog)
      if(xlog.lt.80)runstring(xlog+1:)=' '
```

```
*
*-----< Open the input file.
*
      error=fmpopen(indcb,error,runstring,'or',1)
      if(error.lt.0)go to 9000

*
*-----< Start processing the commands.
*
10    length=Fmpread(indcb,error,inarray,80)
      if(length.eq.-1)go to 10000          ! EOF condition
      if(error.lt.0)go to 9000

*
*-----< Make sure everything is in upper case
*
      call casefold(instring)

*
*-----< Get the command from the input string
*
      call SplitString(instring,command,instring)

*
*-----< Find out what command was issued.
*-----< Commands are:
*-----<               RN - rename a category or function
*-----<               EC - edit a function within a category
*-----<               EX - exit
*


*
*-----< RN command. It has two subcommands.
*-----<               F - Rename a function
*-----<               C - Rename a category
*
      if(command(1:2).eq.'RN')then
          call SplitString(instring,command,instring)
          if(command(1:1).eq.'C')then

*
*-----< We will rename a category. Get the current name and the
*-----< new name.
*
              call Splitstring(instring,OldCategoryName,instring)
              call Splitstring(instring,NewCategoryName,instring)

*
*-----< Use the SECURITY/1000 routine SecRenameCat to perform the
*-----< rename.
*
              call SecRenameCat(OldCatArray,NewCatArray,error)
```

```
            if(error.lt.0)go to 8000
            go to 10                !go get next command
          endif

*
*-----< Check whether or not we are to rename a function.  If not, give
*-----< up and get the next command from the input file.
*


          if(command(1:1).eq.'F')then

*
*-----< We are to rename a function. Get the name of the category
*-----< that contains the function to be renamed.
*

              call SplitString(instring,CategoryName,instring)

*
*------< Get the current function name and the new name for the
*------< function.
*

              call Splitstring(instring,OldFunctionName,instring)
              call Splitstring(instring,NewFunctionName,instring)

*
*-----< Use the  SECURITY/1000 routine SecRenameFnc to perform the
*-----< the function rename.
*

              call SecRenameFnc(OldFncArray,NewFncArray,
     +                          CatArray,error)
              if(error.lt.0)go to 8000
          endif
          go to 10    !go get next command
       endif


       if(command(1:2).eq.'EC')then

*
*-----< We will edit a function. Get the category containing the
*-----< function and the function to be edited.
*

          call SplitString(instring,CategoryName,instring)
          call SplitString(instring,FunctionName,instring)

*
*-----< Get the 4 CPLVs and convert them to binary. If an error
*-----< results during the conversion, the corresponding CPLV will
*-----< default to -1. A cplv value of -1 tells SECURITY/1000 not to
*-----< update the corresponding CPLV already in the tables.
*

          do j=1,4
              call SplitString(instring,CplvValue,instring)
```

```
            cplvarray(j)=DecimalToInt(CplvValue,error)
            if(error.ne.0)CplvArray(j)=-1
         enddo

*
*-----< Use the SECURITY/1000 routine SecEditFunction to perform
*-----< the actual editing operation.
*

         call SecEditFunction(FncArray,CatArray,
     +                        cplvarray,error)
         if(error.lt.0)go to 8000
         go to 10
      endif

*
*-----< See if an Exit command was issued.
*

      if(command(1:2).eq.'EX')go to 10000


*
*-----< We receive this message if an unrecognized command was seen.
*

      write(1,1000)command
1000  format(" Unrecognized command seen ",a6)
      go to 10

*
*-----< We have a SECURITY/1000 error.   Report it and terminate.
*

8000  write(1,8001)error
8001  format(" Security/1000 error: ",i6.6)
      go to 10000

*
*-----< We have an Fmp error.   Report it and terminate.
*

9000  call FmpreportError(error,runstring)

10000 call FmpClose(indcb,error)
      end
```

Example of a Directory Create Program

```
program mkdir
implicit none

*
*------< This program creates FMP directories. The program subjects the
*------< user to a series of security checks before it will create
*------< directories.
*------<
*------< The first check. Can the group with which the user logged on
*------< create directories? The user's logon group is checked to see
*------< whether it is in the category 'CRDGP'. If it is, this test
*------< is passed and test two is applied.
*------<
*------< The second check. Does the user have enough capability to create
*------< a directory? The base function of the function whose name
*------< matches that of the logon group is checked.
*------<
*------< If both tests are passed, control is passed to FmpCreateDir to
*------< perform the actual directory creation. Note that FmpCreateDir
*------< will perform its own security check as defined in the security
*------< tables. See the SECURITY.TBL file for the security definition
*------< of FmpCreateDir.
*

      integer runarray(40),xlog,GroupArray(5)
      integer SecGetMyCplv,ProgCplv,FlagArray(4),error
      integer DecimalToInt,lu,FmpCreateDir

      character runstring*80,GroupName*10,DirectoryName*64
      character LuString*4

      equivalence (runstring,runarray)
      equivalence (GroupName,grouparray)

*
*------< Collect the run string to find out what the user wants to do.
*

      call getst(runarray,-80,xlog)
      if(xlog.lt.80)runstring(xlog+1:)=' '

*
*------< Get the directory name and LU, if suppiled.


      call SplitString(runstring,DirectoryName,runstring)

*
*------< See if a directory was supplied. If not, issue a little help.
*

      if(DirectoryName(1:1).eq.' ')then
          write(1,100)
100       format(" MkDir Usage: MkDir directory[,lu]")
          call exec(6,0)
      endif
```

```
*
*-----< Get the LU, if supplied.
*

      call SplitString(runstring,LuString,runstring)
      if(LuString(1:1).ne.' ')then
          lu=DecimalToInt(LuString,error)
          if(error.ne.0)then
              write(1,100)
              call exec(6,0)
          endif
          else
          lu=0
      endif

*
*-----< Get my capability level. It will be needed for the security
*-----< check.
*

      ProgCplv=SecGetMyCplv()

*
*-----< Get the name of the group with which the user logged on.
*-----< Use the group name as a function name within category "CRDGP" to
*-----< determine what (if anything) the user can do with this utility.
*

      call GpNam(GroupName)

*
*-----< Now find out what the user can do with us.
*-----< Note that all errors are treated as security violations.
*-----< Also, if SecChkCplvNam is supplied with a category or function
*-----< that is not defined, it will return with the FlagArray set to
*-----< indicate that the check passed all functions (base and the three
*-----< subfunctions). It will also set error to POSITIVE 1702. This
*-----< indicates that the category or function could not be found, so
*-----< a default result, successful, was returned.  Failure to find the
*-----< category or function is NOT considered an error condition by
*-----< SecChkCplvNam.
*

      call SecChkCplvNam(GroupArray,6hCRDGP ,ProgCplv,FlagArray,error)
      if(error.lt.0)then
          write(1,1)
1         format(" SECURITY VIOLATION detected.")
          call exec(6,0)
          else
          if(error.eq.1702)then
              write(1,2)
2             format(" MkDir: No access for your group")
              call exec(6,0)
          endif
      endif

*
*-----< We know now that the user's group can create directories. But
*-----< whether the user can create directories can be determined by
```

```
*------< looking at the flag corresponding to the base function.
*
      if(FlagArray(1).lt.0)then
          write(1,1)
          call Exec(6,0)
      endif


*
*------< The type of directory the user can create will be determined
*------< by the CRDIR function of the $FMP category. The FMP subsystem
*------< will do this checking for us when we call the  FmpCreateDir
*------< routine.
*
      error=FmpCreateDir(DirectoryName,lu)
      if(error.lt.0)then
          call FmpreportError(error,DirectoryName)
      endif


      end
```

SEC1000.LIB

SEC1000.LIB is the programmatic interface of the Security/1000 subsystem. The user level subroutines are described here in alphabetical order.

Note that when a parameter is an entry number, it is always zero relative.

SecChangeCplv

SecChangeCplv allows the calling program to change its PROGCPLV during execution.

```
call SecChangeCplv(newplv,error)
error=SecChangeCplv(newplv,error)
integer newcplv, error
```

`newcplv` An integer that is the new PROGCPLV. It ranges from zero to the maximum of USERCPLV and ORGCPLV. If newcplv is negative, PROGCPLV is set to ORGCPLV.

`error` An integer that returns a negative value if there is an error.

SecChkCplvNam

SecChkCplvNam checks a given CPLV against the capability level of a given function to determine whether the CPLV equals or exceeds that function.

```
call SecChkCplvNam(fnc,catid,chkcplv,flgarray,error)
error=SecChkCplvNam(fnc,catid,chkcplv,flagarray,error)
integer fnc(3), catid(3),flagarray(4)
integer chkcplv, error
```

`fnc` An integer for a function, left justified, blank filled, and upshifted whose CPLV is checked against the supplied CPLV.

`catid` An integer for the category ID, left justified, blank filled, and upshifted of the category owning the function in the fnc parameter.

`chkcplv` An integer for the CPLV against which the capability of the function is checked.

`flagarray` A four word integer array containing the CPLV flags. Each flag indicates whether the chkcplv was less than the CPLVs in the security tables. If a flag is <0, chkcplv was less than the corresponding CPLV in the security table, if >0, chkcplv was equal to or greater than the corresponding security table CPLV. Word 1 = base CPLV, Word 2 = sub1 CPLV, Word 3 = sub2 CPLV, Word 4 = sub3 CPLV.

If the function cannot be found, flagarray is set to positive values. ERROR is then set to positive 1702. This indicates that the test passed because the function could not be found. This is not considered an error condition.

`error` An integer that returns a negative value if there is an error. Note that if a positive 1702 is returned, then the function could not be found. This is not considered an error condition.

If the function has less than 3 subfunctions, the flags of the missing subfunctions are set to a positive value.

SecChkCplvNam differs from SecChkCplvNum in that SecChkCplvNam searches the tables for a particular name while SecChkCplvNum searches for a particular number.

SecChkCplvNum

SecChkCplvNum checks a given CPLV against that of a given function to determine whether it equals or exceeds the capability level of that function.

```
call SecChkCplvNum(fncnum,catnum,chkcplv,flagarray,error)
error=SecChkCplvNum(fncnum,catnum,chkcplv,flagarray,error)
integer fncnum, catnum flagarray(4)
integer chkcplv, error
```

`fncnum` An integer for the entry number in the CFT owned by the category pointed to by catnum. It indicates the function to be used in the CPLV check.

`catnum` An integer for the CIT entry number of the category owning the function pointed to by the fncnum parameter.

`chkcplv` An integer for the CPLV to be checked against the function's capability levels.

`flagarray` A four word integer array containing CPLV flags. Each flag indicates whether the chkcplv was less than the CPLV in the security tables. If flag <0, chkcplv less than corresponding security table CPLV, if >0, chkcplv is equal to or greater than security table CPLV. Word 1 = base CPLV, Word 2 = sub1 CPLV, Word 3 = sub2 CPLV, Word 4 = sub3 CPLV.

If the function cannot be found, flagarray is set to positive values. ERROR is then set to positive 1702. This indicates that the test passed because the function could not be found. This is not considered an error condition.

`error` An integer that returns a negative value if there is an error. Note that if a positive 1702 is returned, then the function could not be found. This is not considered an error condition.

If the function has less than 3 subfunctions, the flags of the undefined subfunctions are set to a positive value.

SecEditFunction

SecEditFunction changes the CPLVs for a specified function.

```
call SecEditFunction(fnc,catid,cplvarray,error)
integer catid(3), fnc(3), cplvarray(4)
integer error
```

`catid` A 3 word integer array for the CIT-defined category ID to which the command belongs. The contents of the array must be left justified, upshifted, and blank filled.

`fnc` A 3 word integer array for the function belonging to catid whose CPLV will be changed. The array must be left justified, upshifted, and blank filled.

`cplvarray` A 4 word integer array containing the new CPLV values. Word 1 = new base CPLV, Word 2 = new sub1 CPLV, Word 3 = new sub2 CPLV, Word 4 = new sub3 CPLV. All CPLVs are in the 0-31 range. If a CPLV is -1, the corresponding CPLV in the CFT entry will not be updated.

`error` An integer that returns a negative value if there is an error. Any of the CPLV parameters may be set to -1 to indicate a CPLV that is not to be updated.

SecGenTables

SecGenTables generates a set of security tables.

```
call SecGenTables(inputfile,listfile,outputfile,keepmac,error)
character*64 inputfile, listfile, outputfile, keepmac
integer error
```

`inputfile` A FTN7X character string for the name of the file containing the source of the security tables.

`listfile` A FTN7X character string for the file to which the listing will be written. The file will be created if it does not exist, overwritten if it does.

`outputfile` A FTN7X character string for the file to which the generated security tables will be written. The file will be created if it does not exist, overwritten if it does.

`keepmac` A FTN7X character string for the name of the file to which the generated MACRO/1000 will be written. The file will be created if it does not exist, overwritten if it does. If it is a blank or ASCII 0, the generated MACRO/1000 is not saved.

`error` An integer that returns a negative if there is an error. All FMP error codes have been biased so you can determine the file to which the error code relates. The bias factors are: SOURCE -2000, LIST -3000, OUTPUT -4000, KEEPMAC -5000. To get original error code: FmpError = error + abs(bias).

SecGetCitNam

SecGetCitName retrieves an entry from the CIT via a category name.

```
call SecGetCitNam(catid,entry,number,address,error)
error=SecGetCitNam(catid,entry,number,address,error)
integer catid(3),entry(4)
integer number, address, error
```

`fnc` A 3 word integer array for the function left justified, blank filled, and upshifted, whose entry is to be copied from the CFT.

`catid` A 3 word integer array for the category ID of the program whose entry is wanted.

`entry` A 4 word integer array for a buffer into which the entry is written.

`number` An integer that returns the entry number.

`address` An integer that returns the entry address.

`error` An integer the returns a negative value if there is an error.

SecGetCftNam

SecGetCftNam retrieves an entry from a CFT via a function and category name.

```
call SecGetCftNam(fnc,catid,entry,number,address,error)
error=SecGetCftNam(fnc,catid,entry,number,address,error)
integer fnc(3), catid(3), entry(4)
integer number, address, error
```

`fnc` A 3 word integer array for the function left justified, blank filled, and upshifted, whose entry is to be copied from the CFT.

`catid` A 3 word integer array for the category ID of the program owning the CFT from which the entry will be copied.

`entry` A 7 word integer array for the buffer into which the entry is to be copied.

`number` An integer that returns the number of the entry within the CFT (zero relative).

`address` An integer that returns the address of the entry.

`error` An integer that returns a negative value if there is an error.

SecGetCitNum

SecGetCitNum retrieves an entry from the CIT via an entry number.

```
call SecGetCitNum(number,entry,address,error)
error=SecGetCitNum(number,entry,address,error)
integer number, address, error, entry(4)
```

`number` An integer for the number of the CIT entry to be retrieved. Entries start numbering at zero.

`entry` A 4 word integer array for the buffer in which the entry is written.

`address` An integer that returns the address of the entry.

`error` An integer that returns a negative value if there is an error.

SecGetCftNum

SecGetCftNum retrieves an entry from a CFT.

```
call SecGetCftNum(fncnum,catnum,entry,adr,error)
error=SecGetCftNum(fncnum,catnum,entry,adr,error)
integer fncnum, catnum, adr, error, entry(4)
```

`fncnum` An integer for the number of the CFT entry to be retrieved. CFT entries number from zero.

`catnum` An integer for the number of the CIT entry that points to the CFT to be used for the fncnum parameter.

`entry` A 7 word integer array for the buffer into which the entry will be copied.

`adr` An integer that returns the entry address.

error An integer that returns a negative value if there is an error.

SecGetCplvs

SecGetCplvs retrieves the CPLVs of the calling program.

```
call SecGetCplvs(usercplv,progcplv,rquscplv,orgcplv)
integer usercplv, progcplv, rquscplv, orgcplv
```

`usercplv` An integer representing the USERCPLV of the calling program.

`progcplv` An integer representing the PROGCPLV of the calling program.

`rquscplv` An integer representing the RQUSCPLV of the calling program.

`orgcplv` An integer representing the ORGCPLV of the calling program.

SecGetMyCplv

SecGetMyCplv retrieves the CPLV word from the caller's ID segment.

```
cplv=SecGetMyCplv()
```

`cplv` An integer for the CPLV word from the caller's ID segment. Note that this is the CPLV word and not the cplv field within the CPLV word.

SecGetRqusCplv

SecGetRqusCplv retrieves the RQUSCPLV of the specified program.

```
call SecGetRqusCplv(progname,session,rquscplv)
error=SecGetRqUsCplv(progname,session,rquscplv)
integer progname(3)
integer session, rquscplv
```

`progname` An integer for the program name, left justified, blank filled, and up-shifted, of the ID segment whose RQUSCPLV field is to be retrieved. If the first word of the program is 0, the calling program is assumed to be the program specified.

`session` An integer for the session number to which the program belongs. There are 2 special values for session: -1 = calling program's session, 0 = system session.

`rquscplv` An integer for the RQUSCPLV field from the ID segement that returns a negative if there is an error.

`error` An integer that returns a negative value if there is an error.

SecGetProgCplv

SecGetProgCplv retrieves the PROGCPLV of the specified program.

```
call SecGetProgCplv(progname,session,rquscplv)
error=call SecGetProgCplv(progname,session,progcplv)
integer array progname
integer session, progcplv, error
```

`progname` An integer array for the program name of the ID segment whose RQUSCPLV field is to be retrieved. If the first program word is 0, the calling program is assumed to be the program specified.

`session` An integer for the session number to which the program belongs. There are 2 special values for session: -1 = calling program's session, 0 = system session.

`progcplv` An integer for the PROGCPLV field from the ID segement that returns a negative if there is an error.

If there was no error, the B-register on return will contain the address of the CPLV in the ID segment.

SecInitialize

SecInitialize initializes Security/1000 and turns it on as an option.

```
call SecInitialize(snapfile,onoff,error)
character*(*) snapfile
integer onoff, error
```

`snapfile` A character string for the name of the snapfile created at generation time and used by the current system.

`onoff` An integer flag for on (zero) and off (number other than zero).

`error` An integer that returns a negative value if there is an error.

SecListTables

SecListTables produces a listing of currently installed security tables.

```
call SecListTables(listfile,error)
character*(*) listfile
```

`listfile` A character string representing the file to which the listing will be written. The file is created if it does not exist, overwritten if it does.

`error` An integer that returns a negative value if there is an error.

SecOnOf

SecOnOf determines whether Security/1000 is on or off.

```
flag=SecOnOf(flag) or
flag=SecOnOff(flag) or
flag=SecOnOf() or
flag=SecOnOff()
```

`flag` An integer showing whether Security/1000 is on or off. Flag = 0 (on) and flag = integer that is not zero (off).

SecProgCplv

SecProgCplv retrieves the 3 fields from the CPLV word in the calling program's ID segment. If idsegadr is supplied, the information is retrieved from the specified ID segment.

```
call SecProgCplv(orgcplv,rquscplv,curcplv[,idsegadr])
integer orgcplv, rquscplv, curcplv, idsegadr
```

`orgcplv` An integer that returns the original program CPLV.

`rquscplv` An integer that returns the required user CPLV for the program.

`curcplv` An integer that returns the program's current CPLV.

`idsegadr` An integer for an optional parameter. If supplied, it is the address of the ID segment whose CPLV fields are to be retrieved.

No error checking is done on the ID segment address which is assumed to be correct.

SecPutCitNam

SecPutCitName updates an entry in the CIT via a catid name.

WARNING

This routine does not check whether or not the updated entry will cause duplicate categories or functions. If used incorrectly, this routine will corrupt the CIT.

```
call SecPutCitNam(catid,entry,number,address,error)
error=SecPutCitNam(catid,entry,number,address,error)
integer catid(3), entry(4)
integer number address, error
```

`fnc` An integer for the function, left justified, blank filled, and upshifted, whose entry is to updated.

`catid` An integer for the category ID, left justified, blank filled, and upshifted, of the category whose entry is to be updated.

`entry` A 4 word integer array for a buffer which contains the updated version of the entry.

`number` An integer that returns the number of the entry within the CIF (zero relative).

`address` An integer that returns the address of the entry within CIT.

`error` An integer that returns a negative value if there is an error.

SecPutCftNam

SecPutCftNam updates an entry in a CFT.

WARNING

This routine does not check whether or not the updated entry will cause duplicate categories or functions. If used incorrectly, this routine will corrupt the CFT.

```
call SecPutCftNam(fnc,catid,entry,number,adr,error)
error=SecPutCftNam(fnc,catid,entry,number,adr,error)
integer fnc(3), catid(3), entry(4)
integer number, adr, error
```

`fnc` An integer for the function, left justified, blank filled, and upshifted, whose entry is to be updated.

`catid` An integer for the category ID, left justified, blank filled, and upshifted, of the function owning the CFT to which the entry will be copied.

`entry` A 7 word integer array for the buffer which contains the updated version of the entry.

`number` An integer that returns the number of the entry within the CFT (zero relative).

`adr` An integer that returns the address of the entry within the CIT.

`error` An integer that returns a negative value if there is an error.

Note that this routine does not check whether or not the updated entry will cause duplicate categories or functions.

SecPutCitNum

SecPutCitNum puts an updated entry back in the CIT via an entry number.

WARNING

This routine does not check whether or not the updated entry will cause duplicate categories or functions. If used incorrectly, this routine will corrupt the CIT.

```
call SecPutCitNum(number,entry,address,error)
error=SecPutCitNum(number,entry,address,error)
integer number, address, error, entry(4)
```

`number` An integer that contains the number of the CIT entry to be updated. Entries number from zero.

`entry` An integer for the buffer containing the updated version of the entry. It must be at least 4 words long.

`address` An integer that returns the address of the entry.

`error` An integer that returns a negative value if there is an error.

SecPutCftNum

SecPutCftNum updates and returns the CFT entry to its relevant CFT via the entry numbers.

WARNING

This routine does not check whether or not the updated entry will cause duplicate categories or functions. If used incorrectly, this routine will corrupt the CFT.

```
call SecPutCftNum(fncnum,catnum,entry,adr,error)
error=SecPutCftNum(fncnum,catnum,entry,adr,error)
integer fncnum, catnum, adr, error, entry(7)
```

`fncnum` An integer representing the number of the CFT entry to be updated. CFT entries number from zero.

`catnum` An integer for the number of the CIT entry that points to the CFT to be used by the cmdnum parameter.

`entry` An integer for the buffer containing the updated entry version. It must be at least 7 words long.

`adr` An integer that returns the address of the entry.

`error` An integer that returns a negative value if there is an error.

SecPutRqusCplv

SecPutRqusCplv sets the RQUSCPLV of a program.

```
error=SecPutRqusCplv(progname,session,newrquscplv,error)
integer progname(3)
integer session, newrquscplv, error
```

`progname` An integer for the program name, left justified, blank filled. The program must have an ID segment set up. If the first word of progname is zero, the calling program's name is used.

`session` An integer representing the session number in which the program resides. If the number is negative, use the calling program's session, if 0, the system session.

`newrquscplv`
An integer which is the new RQUSCPLV value. This value cannot ex-

ceed the PROGCPLV of the program from which SecRqusCplv was called. Note that only the 5 lsb are looked at.

`error` An integer that returns a negative value if there is an error.

SecPutProgCplv

SecPutProgCplv sets the PROGCPLV of a program.

```
error=SecPutProgCplv(progname,session,newprogcplv,error)
integer array progname
integer session, newprogcplv, error
```

`progname` An integer for the program name, left justified, blank filled. The program must have an ID segment set up. If the first word of progname is zero, the calling program's name is used.

`session` An integer representing the session number in which the program resides. If the number is negative, use the calling program's session, if 0, the system session.

`newprogcplv`
An integer which is the new PROGCPLV value. This value cannot exceed the PROGCPLV of the program from which SecProgCplv was called. Note that only the 5 lsb are looked at.

`error` An integer that returns a negative value if there is an error.

SecRenameCat

SecRenameCat renames a cat ID in the CIT. The old ID must exist and the new one must not. Note that this routine does not modify the case of the parameters. It is the caller's responsibility to ensure that the data passed to this routine is in the correct case.

```
call SecRenameCat(oldcatid,newcatid,error)
integer oldcatid(3), newcatid(3)
integer error
```

`oldcatid` An integer array representing the old cat ID, left justified, blank filled, and upshifted.

`newcatid` An integer array representing the new cat ID, left justified, blank filled, and upshifted.

`error` An integer that returns a negative value if there is an error.

SecRenameFnc

SecRenameFnc renames a specified function in a specified program. Note that this routine does not modify the case of the parameter. It is the caller's responsibility to ensure that the data passed to this routine is in the correct case.

```
call SecRenameFnc(oldfnc,newfnc,catid,error)
integer array oldfnc, newfnc, catid
integer error
```

`oldfnc` An integer array representing the old function name, left justified blank filled, which MUST exist in the CFT belonging to the catid.

`newfnc` An integer array representing the new function name,left justified blank filled, which MUST NOT exist in the CFT belonging to the catid.

`catid` An integer array that is the category ID owning the function about to be renamed. The catid must be defined in the CIT. Left justified, blank filled, upshifted.

`error` An integer that returns a negative value if there is an error.

SecSwitch

SecSwitch switches Security/1000 on or off.

```
call SecSwitch(switch,error)
integer switch, error
```

`switch` An integer that is the switch directive: 0 = off, not zero = on.

`error` An integer that returns a negative value if there is an error.

SecUserCplv

SecUserCplv retrieves the USERCPLV of the session owning the calling program.

```
Integer SecUserCplv
UserCplv=SecUserCplv()
```

`UserCplv` An integer that is the USERCPLV.

D

Sample Answer File

To install Security/1000, you must modify your answer file. Add the information shown below in boxes to your current answer file:

```
.
.
.
pa cdsfh,time,class,spslg,alarm ,SECOS,CHECK

ms $sysa:92077
end
*
* OS partitions
*
.
.
.
re,SECOS.rel
en
*
re,CHECK.rel
en

.
.
.
*    ***************
*    System Messages
*    ***************
re,%msgtb
en
re,%$m000
en

re,security.rel
en

.
.
.
en
```

```
*    ****************
*    System Libraries
*    ****************
```

lib SEC1000.lib

```
lib bigds.lib
lib fndlb.lib
lib biglb.lib
```

lib SEC1000.lib

```
end
*    ********************
*    CDS System Libraries
*    ********************
```

lib SEC1000CDS.lib

```
lib bgcds.lib
```

lib SEC1000.lib

```
lib bigds.lib
lib fndlb.lib
lib biglb.lib
*
* end
```

E

Logon Files

```
*************************************************************************
**                                                                     **
** LOGON File for James Alabama                                        **
**                                                                     **
** Last changed <870730.1752>                                          **
**                                                                     **
*************************************************************************
*
* First set the logging to the screen to off
SET LOG = OF
* Logoff user after 3 timeouts
SET AUTO_LOGOFF = 3
*
* Set a variable called 'user-name' that can be used to identify user
*
SET USER_NAME = JAMES
*
* Make sure that the stack is saved
*
SET SAVE_STACK = TRUE
*
* Set the prompt to be the user's name
*
SET PROMPT = $USER_NAME>
*
* Set UDSP paths first to working directory. then system. and then
* user dir.
*
PATH 1 ./PROGRAMS /$USER_NAME/PROGRAMS
PATH 2 ./CMDFILE /$USER_NAME/CMDFILES
*
* Echo a friendly message
ECHO 'Hello ' $USER_NAME
RETURN
```

Global Logon File

```
*************************************************************************
**                                                                     **
** GLOBAL_LOGON file for all users                                     **
**                                                                     **
** Last changed <870730.1753>                                          **
**                                                                     **
*************************************************************************
*
```

```
* A message that LU's are being verified or backed up may be written
* here and then an 'EX' can be put in to keep all users off the system.
*
*
* Show the users the system messages.
LI /SYSTEM/SYSTEM_MESSAGE
*
* Other messages or commands can be executed for all users to see
*
* Note that the next command looks into the user's default working
* directory for the personal logon file.
IF DL LOGON.CMD,,0
   THEN TR,LOGON.CMD
FI
*
*
*
```

F

GRUMP Command/Log Files

GRUMP Command File Example

```
*  This is an example of a GRUMP command file.
*  It creates group MAX.
*  Then user DEDRA is created and added to group MAX during
*    the creation of her user account.
*  Group CPE is created.
*  Then DEDRA is added to group CPE by altering her existing
*    user account.
*  DEDRA's user account is given the password "bluejay"
*    during creation and that password is changed to no
*    password during the alteration of her user account.
*  The user and group accounts created are then purged from
*    the multiuser system.
*
*  All lines beginning with an asterisk are comments
*    included to explain what is happening.
*
*  Assumptions made for this file to work:
*       1. User DEDRA does not exist.
*       2. Groups MAX and CPE do not exist.
*       3. Directories /DEDRA and /DEDRA/MAX do not exist.
*       4. The command file name is TREX.
*
*  Usage:    RU,GRUMP,TREX
*
*  If you want to see the corresponding log file, the usage
*       string would be:
*
*              RU,GRUMP,TREX,LOGTREX
*
*  Make sure that file LOGTREX does not exist before
*       specifying it as the log file.
*
*
* >> Create group MAX and provide group information.
*
NEW,GROUP,MAX,4:30:0,500:0,-100:255,/e
*
* >> Create user DEDRA.
*
NEW,USER,DEDRA
*
```

```
* >> Provide unique user information.
*
`Dedra Jackson`,BLUEJAY,25,-200:250,220,225,/E,4:4
*
* >> Provide DEDRA.NOGROUP information.  Directory ::DEDRA
* >> does not exist and we want to create it on LU 17 (which
* >> explains the "YES,17" response).
*
* >> If directory ::DEDRA already exists, the input line
* >> must be
* >> "::DEDRA,`RU CI.RUN::PROGRAMS LOGON.CMD::USERS`".
*
* >> If directory ::DEDRA does not exist and you do not want
* >> to create it, the beginning of the input line must be
* >> "::DEDRA,NO,`RU CI.RUN::PROGRAMS LOGON.CMD::USERS`".
*
* >> Note that ::DEDRA is the default for the working directory
* >> so ",YES,17,`RU CI.RUN::PROGRAMS LOGON.CMD::USERS'"
* >> gives the same result as the line below.
*
* >> If a command extends beyond one line, GRUMP will parse
* >> a parameter-separating comma at the end of a line as an
* >> undefined parameter and result in error.  Omit end of
* >> line commas if they are separators rather than place
* >> holders.  Continue the command in column one of the next
* >> line.
*
::DEDRA,YES,17,`RU CI.RUN::PROGRAMS LOGON.CMD::USERS`
`LOGOF.CMD::DEDRA`,0:20:0,50:0
*
* >> YES we want to add DEDRA to an existing group other
* >> than NOGROUP.
*
YES
*
* >> Provide the group name, MAX, and the DEDRA.MAX information.
*
MAX,/DEDRA/MAX,YES,,`RU CI.RUN::PROGRAMS LOGON.CMD`,,-1,-1
*
* >> YES MAX should be the default logon group.
*
YES
* >> /E says we do not want to add DEDRA to any other
* >> existing groups.
*
/E
*
* >> Create group CPE and provide group information.
*
NEW,GROUP,CPE,0:45:0,400:0,-100:255,/E
*
* >> Alter unique user information for DEDRA so we can add
* >> her to the existing group CPE.
*
* >> Leave all the unique user attributes the same except
* >> change the password which was previously defined, to
* >> no password (this explains the YES).
*
ALTER,USER,DEDRA.,,,,,YES,,,,
```

```
*
* >> YES we want to add DEDRA to an existing group.
*
YES
*
* >> Provide group name and DEDRA.CPE information.  Directory
* >> ::DEDRA already exists at this point so GRUMP does not
* >> ask whether we want to create it and which LU it should
* >> go on.
*
CPE,::DEDRA,`RU CI.RUN::PROGRAMS LOGON.CMD::DEDRA`,`LOGOF.CMD::CPE`
-1,-1
*
* >> No we do not want to add DEDRA to any other existing
* >> groups (this explains the /E).
*
/E
*
* >> CPE should be the default logon group
*
CPE
*
* >> Purge user DEDRA
*
PU,USER,DEDRA,OK
*
* >> Purge group MAX
*
PU,GROUP,MAX,OK
*
* >> Purge group CPE
*
PU,GROUP,CPE,OK
*
* >> Exit GRUMP
*
EX
```

Log File Example

This is an example of a log file. This log file would be created if specified in the RUN command with GRUMP command file immediately preceding.

```
* GRUMP>
NEW
* Enter (G)roup or (U)sers :
GROUP
* Enter group logon name :
MAX
* Creating group MAX.
* Enter CPU limit (hh:mm:ss or -1 for No Limit) [No Limit]:
4:30:0
* Enter connect time limit (hh:mm:ss or -1 for No Limit) [No Limit]:
500:0
* LU access table modifications ( [-]LU#[:LU#2] ) :
-100:255
```

```
* LU access table modifications ( [-]LU#[:LU#2] ) :
/E
* GRUMP>
NEW
* Enter (G)roup or (U)sers :
USER
* Enter user logon name :
DEDRA
* Creating user DEDRA
* Enter users real name [???] :
Dedra Jackson
* Enter password (a <cr> gives no password) :
BLUEJAY
* Enter capability level (31=SU) [10] :
25
* LU access table modifications ( [-]LU#[:LU#2] ) :
-200:250
* LU access table modifications ( [-]LU#[:LU#2] ) :
220
* LU access table modifications ( [-]LU#[:LU#2] ) :
225
* LU access table modifications ( [-]LU#[:LU#2] ) :
/E
* Enter #UDSPs:depth [0:0] :
4:4
*
* All users must be in the group NOGROUP.
*
*
* Enter information for DEDRA.NOGROUP
*
* Enter working directory name [::DEDRA] :
::DEDRA
* Create directory ::DEDRA (Yes/No) [N] :
YES
* What LU should the directory go on [0] :
17
* Enter the startup command [RU CI.RUN::PROGRAMS] :
RU CI.RUN::PROGRAMS LOGON.CMD::USERS
* Enter the logoff program/command file [NOT DEFINED] :
LOGOF.CMD::DEDRA
* Enter CPU limit (hh:mm:ss or -1 for No Limit) [No Limit] :
0:20:0
* Enter connect time limit (hh:mm:ss or -1 for No Limit) [No Limit] :
50:0
*
* Do you wish to include the user in any existing
* group other than NOGROUP (Yes/No) [N] :
YES
*
* Enter group name (/E or <cr> to end) :
MAX
*
* Enter information for DEDRA.MAX.
*
* Enter working directory name [::DEDRA] :
/DEDRA/MAX
* Create directory (Yes/No) [N] :
YES
```

```
* What LU should the directory go on [0] :

* Enter the startup command [RU CI.RUN::PROGRAMS] :
RU CI.RUN::PROGRAMS LOGON.CMD::USERS
* Enter the logoff program/command file [NOT DEFINED] :

* Enter CPU limit (hh:mm:ss or -1 for No Limit) [No Limit] :
-1
* Enter connect time limit (hh:mm:ss or -1 for No Limit) [No Limit] :
-1
* Should this be the default logon group (Yes/No) [N] :
YES
*
* Enter group name (/E or <cr> to end) :
/E
* Group MAX is the default logon group.
* GRUMP>
NEW
* Enter (G)roup or (U)sers :
GROUP
* Enter group logon name :
CPE
* Creating Group CPE.
* Enter CPU limit (hh:mm:ss or -1 for No Limit) [No Limit] :
0:45:0
* Enter connect time limit (hh:mm:ss or -1 for No Limit) [No Limit] :
400:0
* LU access table modifications ( [-]LU#[:LU#2] ) :
-100:255
* LU access table modifications ( [-]LU#[:LU#2] ) :
/E
* GRUMP>
ALTER
* Enter (G)roup or (U)sers :
USER
* Enter user.group parameter:
DEDRA.
* Enter new user logon name [DEDRA] :

* Enter users real name [Dedra Jackson] :

* Enter password (a <cr> gives no password) :

* Change password to no password (Yes/No) [N] :
YES
* Enter capability level (31=SU) [25] :

* LU access table modifications ( [-]LU#[:LU#2] ) :

* Enter #UDSPs:depth [4:4] :

* Modified unique user information.
*
* Do you wish to include the user in any existing
* group other than NOGROUP (Yes/No) [N] :
YES
* Enter group name (/E or <cr> to end) :
CPE
*
```

```
* Enter information for DEDRA.CPE.
*
* Enter working directory name [::DEDRA] :
::DEDRA
* Enter the startup command [RU CI.RUN::PROGRAMS] :
RU CI.RUN::PROGRAMS LOGON.CMD::DEDRA
* Enter the logoff program/command file [NOT DEFINED] :
LOGOF.CMD::CPE
* Enter CPU limit (hh:mm:ss or -1 for No Limit) [] :
-1
* Enter connect time limit (hh:mm:ss or -1 for No Limit) [No Limit] :
-1
*
* Enter group name (/E or <cr> to end) :
/E
*
* Which group should be the default logon group [MAX] :
CPE
* Group CPE is the default logon group.
* GRUMP>
PU
 *Enter (G)roup or (U)sers :
USER
* Enter user[.[group][,OK]] :
DEDRA
OK
* DEDRA.NOGROUP record purged
* DEDRA.MAX record purged
* DEDRA.CPE record purged
* User DEDRA successfully purged.
* GRUMP>
PU
* Enter (G)roup or (U)sers :
GROUP
* Enter group[,OK] :
MAX
OK
* Purged group MAX
* GRUMP>
PU
* Enter (G)roup or (U)sers :
GROUP
* Enter group[,OK] :
CPE
OK
* Purged group CPE.
* GRUMP>
EX
```

Index

H

I

K

L

M

N

P

R

S

T

U

Manual Part No. 92077-90056
Printed in U.S.A. August, 1987
E0887

HEWLETT PACKARD