UNICOS® under UNICOS® Administrator’s Guide

SG–2156 10.0

New Features

UNICOS® under UNICOS® Administrator's Guide SG–2156 10.0

This revision of *UNICOS under UNICOS Administrator's Guide*, Cray Research publication SG–2156, supports the 10.0 release of the UNICOS operating system. It contains the following changes:

- UNICOS under UNICOS does not support CRAY EL series and CRAY J90 series systems with VME IOS hardware.
- References to *Trusted UNICOS* have been changed to *Cray ML-Safe*.
- Information has been added regarding the user's `guest.rc` file not being updated automatically.
- Information has been added on UNICOS under UNICOS interactions with the SWS, GigaRing environment, and SSD-T storage.
- Information has been updated on validating a user.
- Instructions have been updated for configuring links .
- Examples have been updated.

Record of Revision

Version	*Description*
10/94	October 1994 Original Printing. This printing supports UNICOS 7.0.7 and 8.0.3 and provides information for system administrtors for the initial release of the UNICOS under UNICOS feature.
8.0.4	May 1995 This revision supports the UNICOS 8.0.4 release of the UNICOS under UNICOS feature.
9.0	August 1995 This revision supports the UNICOS 9.0 release of the UNICOS under UNICOS feature.
10.0	November 1997 This revision supports the UNICOS 10.0 release of the UNICOS under UNICOS feature.

Contents

Preface

This publication describes the UNICOS under UNICOS feature. It provides the descriptions and procedures necessary to accomplish the following tasks:

- Configure a guest system
- Boot a guest system
- Troubleshoot a guest system

Related publications

The following documents contain additional information that may be helpful:

- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011
- *UNICOS Installation Guide*, Cray Research publication SG–2112
- *NQE Administration*, Cray Research publication SG–2150
- *General UNICOS System Administration*, Cray Research publication SG–2301
- *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303
- *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412
- *UNICOS Installation and Configuration Tool Reference Manual*, Cray Research publication SR–3090

Ordering Cray Research publications

The *User Publications Catalog*, Cray Research publication CP–0099, describes the availability and content of all Cray Research hardware and software documents that are available to customers. Cray Research customers who subscribe to the Cray Inform (CRInform) program can access this information on the CRInform system.

To order a document, either call the Distribution Center in Mendota Heights, Minnesota, at +1–612–683–5907, or send a facsimile of your request to fax number +1–612–452–0141. Cray Research employees may send electronic mail to `orderdsk` (UNIX system users).

Customers who subscribe to the CRInform program can order software release packages electronically by using the `Order Cray Software` option.

Customers outside of the United States and Canada should contact their local service organization for ordering and documentation information.

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers:

1	User commands
1B	User commands ported from BSD
2	System calls
3	Library routines, macros, and opdefs
4	Devices (special files)
4P	Protocols
5	File formats
7	Miscellaneous topics
7D	DWB-related information
8	Administrator commands

Some internal routines (for example, the `_assign_asgcmd_info`() routine) do not have man pages associated with them.

variable	Italic typeface denotes variable entries and words or concepts being defined.

`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.

The following machine naming conventions may be used throughout this document:

Term	Definition
Cray PVP systems	All configurations of Cray parallel vector processing (PVP) systems.
Cray MPP systems	All configurations of the CRAY T3D series. The UNICOS operating system is not supported on CRAY T3E systems. CRAY T3E systems run the UNICOS/mk operating system.
All Cray Research systems	All configurations of Cray PVP and Cray MPP systems that support this release.

The default shell in the UNICOS and UNICOS/mk operating systems, referred to in Cray Research documentation as the *standard shell*, is a version of the Korn shell that conforms to the following standards:

- Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) Standard 1003.2–1992
- X/Open Portability Guide, Issue 4 (XPG4)

The UNICOS and UNICOS/mk operating systems also support the optional use of the C shell.

Cray UNICOS Version 10.0 is an X/Open Base 95 branded product.

Reader comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. You can contact us in any of the following ways:

- Send us electronic mail at the following address:

  `publications@cray.com`

- Contact your customer service representative and ask that an SPR or PV be filed. If filing an SPR, use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.

- Call our Software Publications Group in Eagan, Minnesota, through the Customer Service Call Center, using either of the following numbers:

  1–800–950–2729 (toll free from the United States and Canada)

  +1–612–683–5600

- Send a facsimile of your comments to the attention of "Software Publications Group" in Eagan, Minnesota, at fax number +1–612–683–5599.

We value your comments and will respond to them promptly.

Introduction [1]

The UNICOS under UNICOS feature lets a site run two copies of the UNICOS operating system concurrently on a single Cray Research mainframe. One UNICOS system, the *host*, boots normally with most of the system resources. A second UNICOS system, the *guest*, can be started by an authorized user. This second system should be considered a completely **separate** Cray system. A guest system requires its own administration (separate from the host system) and is seen as an independent machine by users and networks.

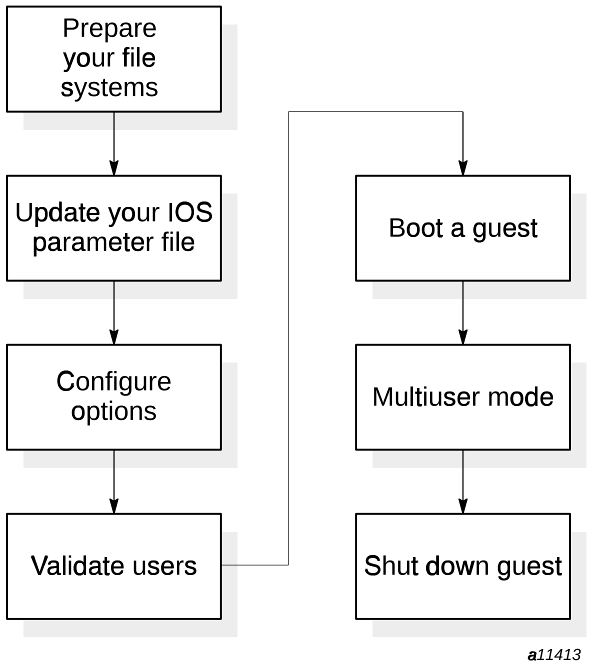
When planning to run or configure a guest system, you must keep in mind that a guest is a complete second operating system. It requires the same minimal resources as any stand-alone kernel, including mainframe memory and basic file systems.

Both the host and the guest system must contain support for the UNICOS under UNICOS feature.

> **Note:** This feature is not an application running on the host. There are two distinct kernels coexisting on the hardware.
>
> The guest feature is designed for software testing and system upgrading, and not for production.
>
> Although this feature is designed for multiple guests, only **one** active guest is currently supported on a host system.

1.1 Checklist

This document is intended to guide you through the entire process of planning, installing, configuring, and running a guest system. The following is a summary of the steps needed to get your guest system running in multiuser mode.



The following is a checklist of changes you need to make to get a guest system running in multiuser mode:

- Prepare your roots for running as a host or a guest

  You will need to change your root file systems so that they can be run as either a host or a guest. This entails creating guest and host versions of selected files (such as `/etc/fstab`).

- Update your IOS parameter to accommodate a guest system and create a guest IOS parameter file (primarily for the SSD solid-state storage device partitioning considerations).

- Configure your UNICOS under UNICOS feature options

  Use the UNICOS Installation/Configuration Menu System to set up your guest system profile (for things like maximum guest memory, CPU percentage, and so on).

- Validate a user

- Boot a guest to single-user mode

- Obtain the guest console and bring a guest to multiuser mode
- Shut down your guest system

1.2 Organization

This document describes the UNICOS under UNICOS feature and is organized as follows:

- Features
- Hardware support
- Software support
- Limitations
- Installation and configuration
- Getting started
- Multiuser mode guests
- Monitoring and analysis
- Troubleshooting
- `guest` Messages

Features [2]

This chapter describes the general capabilities of the UNICOS under UNICOS feature. Figure 1 illustrates the basic concept of the feature.

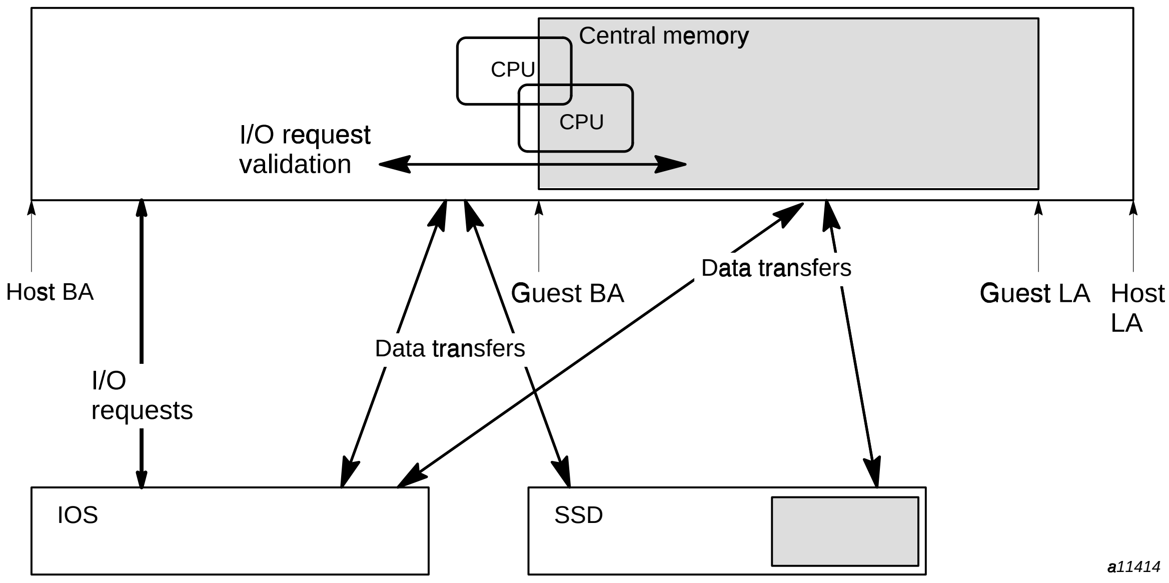


Figure 1. UNICOS under UNICOS concept

At guest boot time, the guest kernel and associated user space are allocated contiguously from the high address end of host user memory. Although the host kernel may access all of the physical memory, the guest kernel is limited to its own system and user space.

Guest central memory (shaded box) allocation is static in nature. That is, guest system memory is allocated dynamically from host memory at guest boot time, and is returned at guest termination. Guest memory is static for the life of the guest.

The administrator can configure minimum host CPUs and the guest user can set the maximum number of CPUs.

The SSD solid-state storage device is also statically allocated. SSD space that you plan to use on the guest must not be in use by the host at guest boot time.

This is also true for guest SSD memory (shaded box). The SSD allocation cannot be altered while the guest is running.

Guest CPU allocation, on the other hand, is dynamic. By default, there are no minimum or maximum number of CPUs that must (or can) be assigned to a guest system. Rather, the host is aware of the user workloads on both host and guest systems, and it will allocate CPUs to satisfy each of the kernel's requirements (within the constraints of CPU load balancing which is specified by the system administrator using the `guest`(1) command).

The IOS is shared by the host and guest systems and is unaware of the guest's presence. IOS code was not needed to support the UNICOS under UNICOS feature. SSD and IOS data transfers occur directly to and from guest memory, rather than being buffered through the host.

The major functional aspects of the UNICOS under UNICOS feature are as follows:

- A system containing support for the guest feature can run as either a guest or a host without recompiling.
- Guest and host kernels can be of different release and revision levels (for example, you can boot a UNICOS 10.0 guest from a UNICOS 9.0.2 host).

  Support will be provided for running a revision (such as UNICOS 9.0. *n*+1) to a particular level as a guest to the preceding revision (UNICOS 9.0.*n*) of that level. Support also will be provided for running one major release level (such as UNICOS 10.0) as a guest to the preceding release level (UNICOS 9.0).

  Major system architectural changes may prevent a new release level from running as a guest to the predecessor system. For a list of releases supporting your kernel as a guest, see the current *Cray Research Service Bulletin* (CRSB) .
- Guest kernels run in monitor mode.
- UNICOS under UNICOS safeguards :
  – Guest kernels can access only the memory allocated to them.
  – The host detects physical disk and SSD slice overlap.
  – The host detects I/O path overlaps.
- The host validates all SSD and IOS requests. Memory transfers are then made directly to and from the guest system's memory.

- The host performs the following functions on behalf of the guest:
  - User exchanges
  - Memory error handling
  - Register parity error handling
  - Physical I/O
- Guest panics typically do not cause a panic in the host.
- CPUs are scheduled dynamically across systems within specified percentages.
- Many major resources (such as SSD, disk, BMR, and network adapters) can be shared .
- The guest and host kernels share the I/O subsystem (IOS). The IOS draws no distinction between guest and host I/O requests.
- The `guest`(1) command is installed in the `/etc` directory. It is the user interface of the UNICOS under UNICOS feature and performs the following functions:
  - Starts a guest system by using the specified amount of memory
  - Stops a guest system
  - Releases guest system memory
  - Changes CPU percentages and ownership of running guests
  - Dumps running or stopped guest system memory
  - Returns the status of all active systems to any user

Hardware Support [3]

This chapter lists the hardware that is supported with the UNICOS under UNICOS feature.

3.1 Mainframes

This feature supports all Cray PVP systems (excluding the CRAY EL series and CRAY J90 series with VME IOS hardware).

Note: CRAY T3E systems running the UNICOS/mk operating system are not supported.

3.2 Hardware requirements

When considering resource requirements, remember that two distinct kernels will reside on your system. The size of each system depends on your specific configurations for each of them, and your requirements for user process space.

UNICOS level	Minimum memory
9.0	8 Mwords
10.0	12 Mwords

Note: A kernel built for a large machine (such as one with 128 Mwords) may not function well (or at all) as a guest on a system with less than 16 Mwords.

3.3 Unsupported hardware

The following peripherals are not supported on your guest system (but are still supported on your host system):

- Expander peripherals
- CRAY T3D systems
- ER90 tape drives
- Cray ND-12 and ND-14 disks
- Maximum Strategy and IBM disk arrays

- Shared file system HIPPI semaphore devices and multiple SSDs
- HIPPI disk and tape devices
- SSD-T (GigaRing Solid State Storage for CRAY J90 and CRAY T90 series systems)

Software Support [4]

This chapter describes the operating system requirements, IOS levels, and supported software for the UNICOS under UNICOS feature.

4.1 Operating system requirements

To use the UNICOS under UNICOS feature for upgrading your UNICOS base system to a new UNICOS operating system, you must meet the following requirements:

- Your host and guest kernels must be one of the following combinations:

Level	9.0 guest	10.0 guest[1]
9.0 host	S	T
10.0 host[1]	S	T

S = Supported

T = Tested and supported

You must have a UNICOS 9.0 (or later) host with a UNICOS 9.0 (or later) guest.

Support information will be kept current in the *Cray Research Service Bulletin* (CRSB).

UNICOS, with the UNICOS under UNICOS feature integrated, was regression tested with the following configurations:

- UNICOS as a stand-alone system
- UNICOS as a host system while running a guest UNICOS system
- UNICOS as a guest system

[1] UNICOS 10.0 is the first release level that supports CRAY J90 and CRAY T90 systems using GigaRing support

4.2 IOS levels

Your IOS level must support the highest level of either your host or guest system. For example, when running UNICOS 10.0 as a guest under UNICOS 9.0.2, your IOS level must be IOS 9.0.2.

4.3 Unsupported software

This feature does not support the following software:

- Operating and monitoring guests from the OWS-E graphical interface (`opi`). Guest system status is **not** shown on the `opi` display.
- Operating and monitoring guests from the SWS (GigaRing based).
- Cray shared file systems (SFSs).

Limitations [5]

This chapter describes hardware and software limitations, as well as limitations on the use of online diagnostics.

5.1 Hardware and software limitations

Although the guest system performance is only marginally less than the host, the UNICOS under UNICOS feature is intended for system testing and upgrading software, and not for creating another production system.

The following are general limitations of this feature:

- The use of a CRAY T3D system from the guest is not supported.
- The guest system is not operated or monitored from the OWS or SWS.
- Tape configurations **must** be identical on the host and the guest. That is, all devices must be in the same position (ordinal) in the configuration. ER90 tapes, though unsupported in the guest, should still be used as place holders in the guest configuration. See Appendix B, page 59, for more information.
- SSD space intended for the guest system must not be in use on the host (that is, SSD space cannot be dynamically allocated at guest boot time). SSD-T storage is not supported (see Section 3.3, page 9).
- Multiple SSDs are not supported.
- Logical CPU 0 is required for booting a guest; therefore, it must not be down at guest boot time.
- Real-time applications may act differently on a guest or a host that has an active guest.
- Dedicating a CPU to a process (`cpu`(8)) in the guest or in a host with an active guest may result in sporadic execution of the process.
- Unlike the NSC N130 network adapter , HIPPI and FCA-1 (FDDI) interfaces cannot be shared between the host and guest for multiple, simultaneous TCP/IP connections.

  Note that a HIPPI channel can still be shared for other applications.

- No GigaRing supported network adapter can be shared. You will need a second network adapter (for example, MPN Ethernet card) for your guest.
- If an alternate path is available for a disk device shared by the host and guest, both paths **must** be configured in each system.

5.2 Online diagnostics

Online diagnostics are available on both the host and the guest. However, it is advisable that you run them only on the host because of the following considerations:

- SMARTE

  System Maintenance and Remote Testing Environment (SMARTE) is only supported on the host.

- CPU diagnostics

  Performance degradation may occur with the CPU diagnostics while the guest is active. The CPU diagnostics program (`olsbt`(8)) should not be run on either the host or the guest while the guest is active.

- OLNET

  When using the NSC module of the OLNET online diagnostic network communications program in the UNICOS under UNICOS environment, the following situation may cause OLNET not to function as expected: when UNICOS configures up an NSC channel, the associated NSC adapter's unit address is sent from the IOS back to UNICOS so that it can verify that the adapter's unit address is what was specified in the `hycf` configuration file. This unit address also is used when sending packets between the IOS and the UNICOS process that requested the transfer.

  The notification of the adapter's unit address is sent to only the UNICOS host or guest that requested that the channel be configured up. If the guest side configured the channel up, the host side would not be notified of the adapter's unit address, and vice versa. What this means to the OLNET NSC test suite is that some data transfer operations may fail when OLNET is run from a UNICOS system that did not configure up the channel. For example, the local loopback (LL) test mode does not operate successfully; however, the remote loopback (RL) test mode can be used to test the local adapter by specifying the local adapter's address as the *remote address* parameter.

Installation and Configuration [6]

This chapter describes installation and configuration procedures for the UNICOS under UNICOS feature. This chapter is only useful for understanding UNICOS under UNICOS configuration issues. Step-by-step instructions are described in Chapter 7, page 23.

6.1 Installation

The UNICOS under UNICOS feature is installed with the base UNICOS product. No special kernel configuration or build is required. To install UNICOS, see the *UNICOS Installation Guide*, Cray Research publication SG–2112.

6.2 Configuration

The following sections describe considerations for configuring a guest system.

6.2.1 Guest feature configuration

The UNICOS under UNICOS feature is configured by using the UNICOS Installation/Configuration Menu System. An example configuration session is shown in Section 7.1. In this section, the term *guest user* refers to any user who has the user database (UDB) privileges required to boot a guest kernel. These privileges are discussed in detail in section Section 7.1.5, page 29.

This menu system performs the following functions:

- Controls disk media access by user
- Converts old structures to new ones
- Imports existing UNICOS under UNICOS configuration information into the menu system configuration database

For this feature, you will need to specify site-specific information in the following areas:

- UNICOS under UNICOS (guest) configuration

  Allows you to configure the options of the UNICOS under UNICOS feature.

- Guest defaults

Sets global default values for all guest users.

- Guest logical device authorization

  Validates users to mount and unmount specific logical disk devices by using the `guest`(1) command.

- Guest file link configuration

  Allows you to specify a list of files that should be linked at multiuser startup to a host or guest counterpart, depending on the running system type.

- Guest user validation

  Allows you to authorize guest users and to further restrict or enhance their capabilities from configured guest defaults.

For information on using the menu system, see the *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412.

6.2.2 File system considerations

This section describes things you should consider when preparing your file systems for use by a guest.

- General information

  Although physical disks can be shared between host and guest, a disk or SSD slice opened on the host cannot be opened on the guest. The host ensures that this requirement is met. An attempt to do so causes an open error on the guest.

  Your `/tmp` and `/swap` space will need to be sufficient to support whatever size load you intend to run on the guest.

  Note: Nonsecure file systems cannot be mounted under a UNICOS multilevel security (MLS) host or guest. Nonsecure file systems may be network file system (NFS) mounted, and in this case, the file systems are restricted to one security label that is defined in the network access list (NAL) entry for the NFS host. If a process is executing with a security label that is unequal to that of the NFS server, the MLS system mandatory access control (MAC) policy will prevent write operations to the NFS-mounted files.

- Guest systems require the following file systems:

  ```
  root
  usr
  swap
  ```

  Typically, `/swap` space can be allocated from unused (available) disk or SSD space. For information on restrictions on SSD usage in a guest, see Section 7.1.1, page 23

- You are advised to have a `/tmp` file system.

- To minimize the impact of having a guest environment coexisting on your hardware platform, you should consider how you want to divide and configure your resources (because a backup or build root file system probably will be used on a guest).

  Traditionally, it has been recommended that you keep your backup root (and backup `/usr`) on a separate device. However, to minimize channel contention and to maintain peak performance when your guest is running, you should consider dividing your host and guest file systems, as shown in Figure 2.

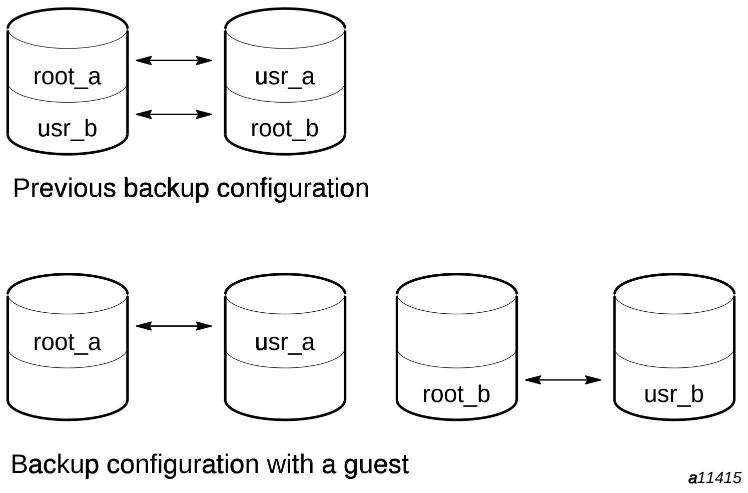

root_a
usr_b
usr_a
root_b
Previous backup configuration
root_a
usr_a
root_b
usr_b
Backup configuration with a guest
a11415

Figure 2. File system disk location

- When mounting the home or other file system(s) for use on a guest, you may consider NFS mounting the host's user file system(s). This provides users immediate access to their existing directories and files.

  Note: If you choose this method for home or other file systems, be sure that users understand that the file system is NFS mounted and is not a virtual copy. All changes will take effect on the host file system. Applications and scripts that usually write to these directories may appear to have failed when run simultaneously on the host and the guest. To avoid this, work in the `$TMPDIR` directory, write output files to `$TMPDIR` (which is local to each system), or append the system's `uts` node name (obtained with `uname -n`) to output file names.

  The user `root` does not have any special permission on an NFS-mounted file system unless you specify: `root=`*hostname* in the `/etc/exports` file. See the `exportfs`(8) man page for more information. If you do not give the guest system NFS-root permission on the host, programs that run as `root` and attempt to create or change files on these NFS file systems will exit abnormally. This can be avoided by running in the `$TMPDIR` directory as well.

6.2.3 IOS parameter files

Before running a guest, you may have to make some minor modifications to your current system's parameter file. The changes are discussed using excerpts from the parameter files. Changes also can be managed by using the UNICOS Installation/Configuration Menu System.

For complete parameter file information and examples, see Section 7.1.3, page 24.

6.2.4 MLS considerations

When a UNICOS multilevel security (MLS) guest runs under a non-MLS UNICOS host, or a non-MLS UNICOS guest runs under a UNICOS MLS host, the full protection of UNICOS MLS is not available. Users of the non-MLS component may be able to gain access to resources controlled by the MLS component, thereby compromising MLS security policies. Although such a configuration is suitable for experimental use of UNICOS MLS, it is not recommended for an MLS production environment.

In a mixed MLS/non-MLS environment, certain file systems are labeled for use under UNICOS MLS, and other file systems are not. Unless a file system is labeled, UNICOS MLS will reject attempts to mount it.

Under non-MLS UNICOS, a secure file system may be mounted only if the `SECURE_MOUNT` configuration option is disabled. Note that MLS attributes of files are not maintained by non-MLS UNICOS, nor are the MLS security policies enforced. As a result, the security of MLS file systems may be compromised if they are mounted under non-MLS UNICOS.

When a UNICOS MLS guest is run with a UNICOS MLS host, the security protections for file systems and data are maintained. However, this configuration is not part of the Cray ML-Safe system configuration.

6.2.5 Configuration links

The UNICOS under UNICOS feature allows you to specify a list of files that should be linked at multiuser startup to host and guest counterparts (such as `.host` or `.guest`) depending on the running system type. It is generally best to configure **all** root file systems on your system to run as both a host or a guest. But when getting started, the configuration links are only required on the root file system, which will be run as a guest.

Although guest and host versions of selected configuration files are not necessary, they do facilitate the use of a single root as either a host or a guest root. By doing this you will facilitate future installations. Take, for example, `/etc/fstab`. Because a guest is not allowed to access physical disk slices mounted on the host, retaining the same `fstab`(5) file (for both host and guest) will result in a significant number of open errors at guest multiuser startup. To avoid such errors, you should create multiple versions of the `/etc/fstab` file so that only the appropriate file systems are mounted when the root is run as either a guest or a host.

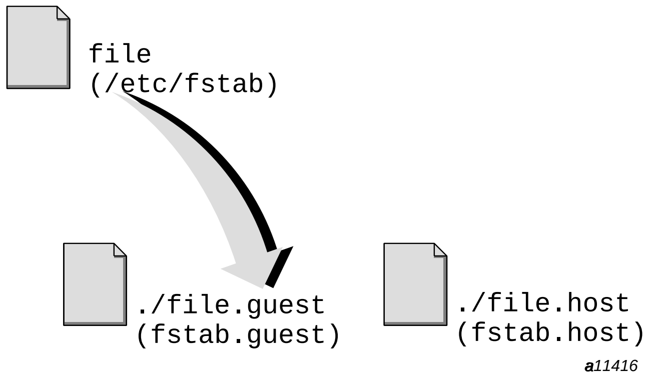


Figure 3. Example link for a system running as a guest

Note: These steps are instructional in nature and **should not** be executed at this time.

To create this link, take the following steps:

1. Invoke the UNICOS Installation/ Configuration Menu System by entering the following commands:

   ```
   cd /mntl/etc/install
   ./install
   ```

2. These commands get you into the main installation menu. Select the following from the menu:

   ```
   M-> Configure System
   ```

3. Select the following option:

   ```
         Asynchronous libraries configuration ==>
         Dumpsys utility configuration ==>
   M->   UNICOS under UNICOS (guest) configuration ==>
         Miscellaneous software configuration ==>
         Import the configuration ...
         Activate the configuration ...
   ```

4. Add `/etc/fstab` to the list of guest configuration files in the `Guest File Link Configuration` menu.

5. Activate the guest configuration.

6. Select and execute `Create Guest and Host versions of files`. This creates the `/etc/fstab.host` and `/etc/fstab.guest` files.

7. Go back to the `Activation Utility` menu, select `Activation Options`, and then select `Activate host or guest versions`. Toggle the selection to `guest` as shown:

```
.   Utilities
.   .   Activation Utility
.   .   .   Activation Options
     Activation root mount poin

     Stop activation on error?                        YES
S->  Activate host or guest versions                  guest
     Activate host or guest versions.
```

8. Go back to the `Import Utility` menu, select `Import Options`, and then select `Import host or guest versions`. Toggle the selection to `guest` as shown:

```
.   Utilities
.   .   Import Utility
.   .   .   Import Options

     Import root mount point                              /
     Stop import on error?                                NO?
S-> Import host or guest versions?                      guest
Reload default import table ...
Activate host or guest versions.
```

9. To import the `fstab` configuration by using the UNICOS Installation/Configuration Menu System, enter the following update screen:

```
.   Configure System
.   .   File System (fstab) Configuration
```

   You are now ready to edit the guest `fstab` information.

10. Remove references to the file systems not needed on the guest system, and activate the `fstab`(5) configuration.

    With the `guest` attribute set in the activation utility, files listed in `/etc/config/guest_config` will be linked to their `.guest` counterparts before the activation. Do not be alarmed if the activation says that it is activating to `/etc/fstab` if your intent is to update `/etc/fstab.guest`.

When a system is booted on this root, / `etc/brc.guest` determines the system type (host or guest) and appropriately sets the soft file links for all files listed in `/etc/config/guest_config.`

11. Return to host `Import/Activate` mode by repeating steps 7 and 8. Choose `host` instead of `guest`.

Getting Started [7]

This chapter describes the steps you need to take to get a guest system up-and-running in single-user mode. It is assumed that you have read the previous chapters of this document. If you want to go immediately to multiuser mode, read this chapter and then proceed to Chapter 8, page 45, which describes additional considerations for parameter and configuration files.

7.1 Preparing your host for a guest

This section describes the items that need to be in place or performed prior to booting a guest system.

7.1.1 General considerations

The following is a list of general considerations:

- Be certain that both your host and guest systems contain the UNICOS under UNICOS feature code. For information about supported systems, see Chapter 3, page 9, and Chapter 4, page 11.
- Evaluate your resource requirements

  General UNICOS System Administration, Cray Research publication SG–2301, contains information about main memory and mass storage requirements for the products associated with a particular UNICOS release. These requirements will vary, depending on the environment you configure for your site.
- Note that IOS-host-guest dumps can be much larger than host dumps (up to twice the size), so you may need to increase your dump partition. If taking dumps to an SWS, be sure that the file system on which dumps are placed has as much available space as possible (remove or compress old dumps).
- Evaluate mainframe memory requirements

  If you intend to run significant loads on both your host and guest systems and your current system is saturated, you may need additional memory. Memory is allocated at guest boot time and requires no changes to the host parameter file. The guest parameter file must not contain the `MEMORY` line.

Keep in mind that the amount of `LDCHCORE` (amount of mainframe memory used for `ldcache`) configured into your kernel or specified in the parameter file may be inappropriate (too large) for your guest system.

- Evaluate SSD requirements

  Unlike main memory, SSD is not dynamically allocated at guest boot time. If you want to allow the use of secondary data segments (SDS) and/or `ldcache` on the guest system, you will have to decrease the allocation of SSD at host boot time. This requires a change in your parameter file (`param`).

  You can still make use of the SSD slices on the host until the guest starts by maintaining them as scratch file systems. Make sure that these devices are not mounted (open) on the host at guest boot time.

  Note: SSD-T storage is not supported on the CRAY J90 and CRAY T90 series of systems.

- Check `CPUS` parameter

  The guest parameter file must not contain the `CPUS` parameter.

7.1.2 Guest file systems

When you are preparing your guest file systems, note the following information:

- A guest requires full root (`/`), `/usr`, and swap file system resources. A separate `/tmp` file system also is suggested. If you do not want to use your current back-up file systems, you will need a new guest root (`/`) and `/usr` file system.
- Any disk slice opened and/or mounted on a guest must **not** be opened and/or mounted on the host. Doing so will result in an error at device-open time.

7.1.3 IOS parameter files

This section describes IOS parameter file considerations.

Before running a guest, you may have to make some minor modifications to your current system's parameter file. The changes are discussed using excerpts from the parameter files. Changes also can be managed by the UNICOS Installation/Configuration Menu System.

Note: Although guest parameter files can be placed anywhere on the file system, it is recommended that you create a directory in `/usr/guest` to hold them.

Depending upon the number of physical slices comprising your guest's locally mounted (that is, non-NFS mounted) file systems, it may be necessary for you to add a `NGRT =` *nnn* specification to your host parameter file. This directive sets the number of GRT (Global Resource table) entries to the specified value. The GRT is present only in the host system.

In general, if the number of file system slices in the guest configuration exceeds one half that of the host, it would be advisable to increase the GRT count. The host's default GRT allocation will accommodate a guest having half the number of physical slices of the host. This should be sufficient for most sites.

If you decide to increase the GRT count, the following formula may be useful:

`NGRT =` *host_slices* `+` *guest_slices* `+` *N* `+` *T*

The factors in the expression are defined as follows:

host_slices	The number of disk and SSD physical slices configured in the host.
guest_slices	The number of disk and SSD physical slices configured in the guest.
N	The total number of network device logical paths configured in the guest.
T	The number of tape units configured in the host.

The host must be rebooted for the GRT size increase to take effect.

7.1.3.1 UNICOS (host) parameter files

The following are considerations for UNICOS (host) parameter files:

- Do you want to allocate SSD memory to the guest? Unlike mainframe memory, SSD memory cannot be allocated at guest boot time. Therefore, if you want the guest to use part of the SSD, you must change the host's IOS parameter file, removing usage of that portion of the SSD to be used by the guest, and reboot the host with the changed IOS parameter file.

  The following IOS model E example reserves two slices of SSD for the guest and requires that the SSD be repartitioned to create another slice.

The following is an SSD configuration before repartitioning:

```
SSD  ssd      {
length 128 Mwords;
pdd ssd_blk0     {minor    2; block        0; length  32768 blocks;}
pdd ssd_blk1     {minor    3; block    32768; length  32768 blocks;}
pdd ssd_blk2     {minor    4; block    65536; length  65536 blocks;}
pdd ssd_blk3     {minor    5; block   131072; length 131072 blocks;}
     }


ldd ssd_blk0     { minor    2; pdd ssd_blk0    ;}
ldd ssd_blk1     { minor    3; pdd ssd_blk1    ;}
ldd ssd_blk2     { minor    4; pdd ssd_blk2    ;}
ldd ssd_blk3     { minor    5; pdd ssd_blk3    ;}
```

The following is a sample configuration after making another slice available for the guest:

```
SSD  ssd      {
length 128 Mwords;
pdd ssd_blk0     {minor    2; block        0; length  32768 blocks;}
pdd ssd_blk1     {minor    3; block    32768; length  32768 blocks;}
pdd ssd_blk2     {minor    4; block    65536; length  32768 blocks;}
pdd ssd_blk2a    {minor    5; block    98304; length  32768 blocks;}
pdd ssd_blk3     {minor    6; block   131072; length 131072 blocks;}
     }


ldd ssd_blk0     { minor    2; pdd ssd_blk0    ;}
ldd ssd_blk1     { minor    3; pdd ssd_blk1    ;}
ldd ssd_blk2     { minor    4; pdd ssd_blk2    ;}
ldd ssd_blk2a    { minor    5; pdd ssd_blk2a   ;}
ldd ssd_blk3     { minor    6; pdd ssd_blk3    ;}
```

The following SSD slices are used on the host:

`ssd_blk0`	For `sdsdev`
`ssd_blk2`	For `swapdev`
`ssd_blk3`	For `/tmp`

The following slices are used by the guest:

`ssd_blk1`	For `swapdev`

`ssd_blk2a`	For `sdsdevw`

- Will the guest be attempting to access a device not usually used on the host? If so, it should be configured for the host as well. Whenever possible, the host should be aware of all hardware to be used by both systems.

  In the following example, the host uses an NSC A130 for its network connection. The guest uses an NSC N130. The N130 could also have been shared between the host and guest by specifying different logical paths in `/etc/hycf.hyp`.

For example, logical path 5 has traditionally been used for TCP/IP on an N130. The adapter address for a system named `sn1703` is `0x86`. The current connection through logical path 5 was specified in `/etc/hycf.hyp`, as follows:

```
direct  sn1703-hyp    8605    ff00    0   16432;
```

The following line was added to allow another connection using logical path 7 for the guest:

```
direct  sn1703a-hyp  8607    ff00    0    16432;
```

Then the following line was added to the `/etc/hosts` file:

```
128.162.82.80   sn1703a sn1703a-hyp sn1703a.cray.com
```

As with any network connection information, these changes must be broadcasted to all machines that require access to the guest.

Note: The N130 and A130 adapter must be made aware of the new IP address by placing the address and its associated logical path in the NSC startup file. See your local network administrator or your NSC service representative for more information.

The network device configuration follows:

```
npdev 0 {
        iopath {cluster 0; eiop 0; channel 030;}
        np_spec S_FEI3FY;
        }
npdev 1 {
        iopath {cluster 0; eiop 0; channel 032;}
        np_spec S_N130;
}
npdev 2 {
        iopath {cluster 0; eiop 0; channel 034;}
        np_spec S_FEIVA;
}
npdev 3 {
        iopath {cluster 0; eiop 0; channel 036;}
        np_spec S_A130;
}
```

7.1.3.2 Guest parameter files

Like any UNICOS system, the guest requires a parameter file. Unlike a stand-alone system, the guest parameter file is not transferred to the operator workstation (OWS or SWS), but it remains on your mainframe disk. To prevent configuration overlaps, make a copy of the host parameter file on the mainframe.

To create a guest parameter file from a copy of the (new) host parameter file created in the previous section, make the following changes:

- Remove the following mainframe specification from the guest parameter file:

  nn `Mwords memory;`

  A guest can be allocated different amounts of memory at each boot. This parameter prevents a guest from starting with anything less than *nn* Mwords.

- Change `rootdev`, `swapdev`, and `sdsdev`:

Old (host):	`rootdev` is `ldd root.a;`
	`swapdev` is `ldd ssd_blk2;`
	`sdsdev` is `pdd ssd_blk0;`
New (guest):	`rootdev` is `ldd root.b;`
	`swapdev` is `ldd ssd_blk1;`
	`sdsdev` is `pdd ssd_blk2a;`

Note: A guest root device mounted on the host at guest boot time will result in a guest system panic. This is due to a configuration overlap. File systems can be automatically unmounted at guest boot time by specifying the corresponding logical device in your guest.rc configuration file. For more information, see the `guest`(1) man page.

7.1.4 Booting the host system

After you edit your parameter file (as described in Section 6.2, page 15), you will need to reboot your system by using the new parameter file in order for the updated information to take effect.

7.1.5 Configuring the UNICOS under UNICOS feature

After you have considered the impact of using this feature on your host, you are ready to configure guest defaults for your system.

During the UNICOS installation procedure, a `/usr/guest` directory is created. The UNICOS under UNICOS feature configuration information is kept in that directory. Although it is not necessary, it is suggested that you make this a separate file system that can be mounted on any root when that root is running as the host.

This section describes how to use the UNICOS Installation/Configuration Menu System to update the guest configuration files. Excerpts from the resulting files are included if you choose to view or edit them manually. If you do edit them manually, you need to set the `GUEST` and/or `GUESTADMIN` permission bits in the users' `udb` entry of each validated user so that the user has the `/etc/guest`(1) command privileges required for booting their guest.

For information on guest permission under a UNICOS MLS system, see the `guest`(1) man page.

1. Invoke the UNICOS Installation/ Configuration Menu System by entering the following commands:

   ```
   cd /etc/install
   ./install
   ```

2. These commands get you into the main installation menu. Select the following from the menu:

   ```
   M-> Configure System
   ```

3. Select the following option:

   ```
         Asynchronous libraries configuration ==>
         Dumpsys utility configuration ==>
   M->   UNICOS under UNICOS (guest) configuration ==>
         Miscellaneous software configuration ==>
         Import the configuration ...
         Activate the configuration ...
   ```

From there, the `UNICOS under UNICOS (guest) Configuration` menu comes up. Select the following from the menu:

```
M-> Guest defaults
```

Initially, you will have defaults shown in the following screen example. In most cases, these defaults should be sufficient. If you want to make changes, refer to the context-sensitive help facility available with the UNICOS Installation/Configuration Menu System.

To see your location in the menu system, use the `Where am I` (`W`) option. The `Guest Defaults` location is shown as follows:

```
    UNICOS 7.0 Installation / Configuration Menu System
.   Configure System
.   .   UNICOS under UNICOS (guest) Configuration
.   .   .   Guest Defaults
```

In this section, the location of menu selections is prefaced with a similar location header.

```
                    Guest Defaults


               Miscellaneous Options

     Maximum number of guest systems allowed.
       A value of zero (0) will DISABLE all
       guest(8) command functions.                  1

     Enable GUEST/HOST kernel tracing?              NO
     Halt HOST when guest panics?                   NO
     Maximum TTY connnections per guest             1
     Create user directories below /usr/guest?      YES
     Remove non-valid directories in /usr/guest     NO

               Memory Size Options

     Minimum HOST memory in megawords               8
     Minimum GUEST memory in megawords              8
     Maximum GUEST memory in megawords              8

               CPU Percentage Options

     CPU allocation scheme                          MEMORY
       Minimum HOST CPU percentage
       Maximum GUEST CPU percentage

 A-> Reset global guest defaults...
```

Upon activation, the information that you provided in the previous menu produces a file called `Defaults` in the `/usr/guest` directory. A sample is included in the following screen.

```
#
#         UNICOS under UNICOS (guest) Global Defaults
#
# This file contains global guest defaults, which may be
# overridden by user values set in the guest Users file.
#
# format:
#       'option' = 'value'
#
#
# Miscellaneous Options
#
MAX_GUESTS=1
GUEST_KERNEL_TRACING=NO
HALT_HOST_ON_GUEST_PANIC=NO
TTY_CONNECTIONS=1
#
# Memory Options
#
MIN_HOST_MEMORY=8
MIN_GUEST_MEMORY=8
MAX_GUEST_MEMORY=8
#
# CPU Percentage Options
#
# CPU_ALLOCATION_SCHEME=MEMORY
```

If your system is significantly larger than the 16 Mwords (such as 128 Mwords), it is probable that the kernel configured for your system will not boot successfully in 8 Mwords. Increase the maximum/minimum guest memory size to at least 16 Mwords and try again.

7.2 Validating a user

After you have configured the guest defaults on the host, a user can be validated to start guest systems. All users, including the administrator, must be validated before they may use the UNICOS under UNICOS feature for anything other than status. Usually, each valid user has a directory in the `/usr/guest` directory.

Note: On an MLS system, you will need to have the `SYSADM` permission (or greater) to activate your configuration.

To validate a user, use the UNICOS Installation/Configuration Menu System, and make the following menu selection:

```
    UNICOS 10.0 Installation / Configuration Menu System
.   Configure System
.   .   UNICOS under UNICOS (guest) Configuration
.   .   .   Guest User Validation
```

After making the selection, press the letter `N`. The following screen appears:

```
    User    Admin?  Guests  Memory  KTrace?  Halt?  Ttys
    -----   -----  ------  ------  ------  ------- -----
E->          NO      0       8        NO      NO    1
```

The minimum requirement for validating a single user is to enter in the user's login name and change the number of allowable guests to 1, as follows. Use the editing capabilities available with the UNICOS Installation/Configuration Menu System to validate users, as shown in the following screen:

```
    User    Admin?  Guests  Memory  KTrace?  Halt?  Ttys
    -----   -----  ------  ------  ------  ------- -----
E->  jqp     NO      1       8        NO      NO    1
```

Note: After you have made any changes, be sure to activate your guest configuration. Also, remind the validated user or users to log out and log back in for UDB changes to take effect.

The information that you provide in this menu updates the `Users` file in the `/usr/guest` directory and in the `/etc/udb` file; it also creates a directory for the user in the `/usr/guest` directory. The `Users` file contains the default settings for each validated user. A skeleton `guest.rc` file is created in the user's `/usr/guest` directory. Changes to this file are required by the user before they can successfully boot a guest. This file is analogous to a user's `.login` or `.profile` file and is not updated by future menu system activations.

7.2.1 Device validation

Now that there is at least one validated user on your system, you must identify the file systems (root (`/`), `/usr`, `/src`) that will be mounted on the guest. If these file systems are usually mounted on the host, guest users must be validated to unmount them at guest system startup.

To validate a user, make the following menu selection:

```
    UNICOS 7.0 Installation / Configuration Menu System
.   Configure System
.   .   UNICOS under UNICOS (guest) Configuration
.   .   .   Guest Logical Device Authorization
```

In the following example, user `jqp` has been validated for `root_d`, `usr_d`, and `src_d`:

```
Guest Logical Device Authorization

      Logical Device     Guest User
       -----------------  ----------
  E->  /dev/dsk/root_d    jqp
       /dev/dsk/usr_d     jqp
       /dev/dsk/src_d     jqp
```

Note: After you have made any changes, be sure to activate your guest configuration.

The following is an example of a resulting configuration file `/usr/guest`:

```
#         UNICOS under UNICOS (guest) User Defaults
#
# This file contains defaults specific to each authorized guest user.

# Options not specified here will default to the settings in
# /usr/guest/Defaults.
#
# format:
#       [user:[user:...]]'option' = 'value'
#
#
# Guest Options for User: jqp
#
jqp:MAX_GUESTS=1
jqp:MAX_GUEST_MEMORY=8
jqp:GUEST_KERNEL_TRACING=YES
jqp:HALT_HOST_ON_GUEST_PANIC=NO
jqp:TTY_CONNECTIONS=1

#
# Logical Devices for User: jqp
#
jqp:LOGICAL_DEVICES="/dev/dsk/root_d \
        /dev/dsk/src_d \
        /dev/dsk/usr_d"
```

Keep in mind that this also will update the UDB when activated.

Note: The user's `guest.rc` file will not be updated by the install tool. It is created for the user as a courtesy when the user is first validated. After that, the user has complete control over his or her `guest.rc` file. If additional file systems are added to the user's authorized list by the administrator, the user will be responsible for appropriately updating his or her `guest.rc` file.

7.2.2 `guest.rc` file

Once validated, you will have a file called `guest.rc` located in the `/usr/guest/` *login* directory.

Note: This file is created for the user only once by the initial activation, and then must be maintained manually by the user. Further install tool activations will not affect the user's `guest.rc` file.

The `guest.rc` file identifies the particular support files and program levels to use for your guest. Review this file, and ensure that the UNICOS version you plan to boot contains support for the UNICOS under UNICOS feature. Pay particular attention to the following entries:

- `NODE_NAME`

  Once again, your guest is a unique system that must have a unique node name. If your kernel was not built with a `uts` node name that is different than that of the host, you need to specify a name here.

- `CRASH`

  Your `crash`(8) level should match the level of your guest kernel.

- `KCOMPRESS`

  Your `kcompress`(8) level **must** match the level of your guest kernel.

- `PARAM`

  Designates the UNICOS parameter file to use for your guest.

- `PARAM_CHECKER`

  Designates the parameter file validation program appropriate for your system level (such as `/etc/econfig`).

- `LOGICAL_DEVICES`

  Denotes the list of logical devices that will be dedicated to your guest and that must be unmounted from the host at guest startup. Review the `/usr/guest/Users` file for your authorized list.

A `guest.rc` file is not required, but it is advisable to always have one.

The following screen shows an example `guest.rc` file for user `jqp`.

```
# This file provides a mechanism for configuring the actions of the guest(1)
# command.  Each available user option is listed and briefly explained. (See
# the guest(1) command man page for more detailed information.)
#
# The option (and comment) format are similar to UNIX shell scripts:
#       option = ’value’
#
# Available options include:
#
#                        Description/
# Option                 (Default)
#----------------------------------------------------
# CRASH                  location of crash binary
#                        (/etc/crash)
#
# DUMP_DIRECTORY         directory below which systems dumps are written
#                        (./dump)
#
# KCOMPRESS              location of kernel (de)compression binary
#                        (/etc/kcompress)
#
# KERNEL                 location of kernel binary
#                        (./unicos)
#
# LOGICAL_DEVICES        list of logical devices to unmount before startup
#                        (NO DEFAULT)
#
# MEMSIZE              requested memory size
#                      (value of MAX_GUEST_MEMORY from
#                        /usr/guest/Users)
#
# MIN_MEMSIZE          minimum memory size that you will accept
#                      (value of MIN_GUEST_MEMORY from
#                       /usr/guest/Defaults)
#
# NODE_NAME            node name
#                      (system name from the kernel binary)
#
# PARAM                location of system parameter file (./param)
#
# TTY_CONNECTIONS      number of tty connections requested (1)
#
```

```
#
# The administrator has preset your options as follows.  (See the
# /usr/guest/Defaults and /usr/guest/Users files for more information on
# your validation.)
#
NODE_NAME=galeg1
PARAM=/usr/guest/ccn/p.root_j
CRASH=/etc/crash
KCOMPRESS=/usr/guest/jqp/kcompress_90
KERNEL=/usr/guest/jqp/unicos_90
LOGICAL_DEVICES="/dev/dsk/root_d \n        /dev/dsk/src_d \n        /dev/dsk/usr_d"
MEMSIZE=8
MIN_MEMSIZE=4
```

7.2.3 Checking your configuration

It is recommended that you verify that the configuration in your `guest.rc` file is valid. To do so, simply use the `-a` option of the `guest`(1) command as follows:

```
guest -a
```

For more information about the `-a` option, see the `guest`(1) man page. Before continuing, correct any errors or warnings noted by this verification option.

7.3 Booting a guest to single-user mode

Now that you have verified your configuration, you are ready to boot your guest system.

Note: Before starting your guest, you should make sure that zip lines other than `0` (the host console) are not in use. The guest boot causes a disconnect to occur on any active zip lines that are needed by the guest.

You should also review your `guest.rc` file to determine whether locations of files (such as `crash`, `kcompress`, and `unicos`) are available in the specified locations. Run the command `guest -a` to authenticate your `guest.rc` configuration and address warnings or errors before proceeding.

7.3.1 Startup

To start your guest operating system, use the following command:

```
guest -s
```

7.3.2 Console connections

If successfully started , the `guest`(1) command reports the number of your guest system's console on the OWS or SWS. Log in to the OWS, and enter the following command:

```
zip #
```

On GigaRing based systems, log in to the SWS and enter the following command:

```
mfcon -l #
```

7.3.3 Start-up output example

The following is an example output for a guest start-up session:

```
gale jqp &lt;23> pwd
/usr/guest/jqp

gale jqp &lt;24> ls
crash_90      guest.rc      map_90
dumps         kcompress_90  unicos_90

gale jqp &lt;25> tail -14 guest.rc
#
NODE_NAME=galeg1

CRASH=./crash_90
KCOMPRESS=./kcompress_90
KERNEL=./unicos_90

PARAM=/usr/guest/ccn/p.root_j
LOGICAL_DEVICES=" /dev/dsk/root_j \
        /dev/dsk/src_j \
        /dev/dsk/usr_j "

MEMSIZE=8
MIN_MEMSIZE=8
```

```
gale jqp &lt;26> guest -s
gst-45 guest: Info
   The guest kernel binary appears to be compressed.
   Decompressing it with:
        ./kcompress_90

gst-46 guest: Info
   The guest kernel binary decompression was successfully completed.

gst-43 guest: Info
   Changing the guest system name from:
        gale
   to:
        galeg1
/gst90/usr/src is mounted.
        Attempting to unmount...done.

/gst90/usr is mounted.
        Attempting to unmount...done.

/gst90 is mounted.
        Attempting to unmount...done.

gst-53 guest: Info
   A 8 MW Guest system (galeg1) has been successfully started for
user:
        jqp

gst-55 guest: Info
   The guest (galeg1) system console is:
        zip 1
   on the host (gale) system’s OWS.
```

7.3.4 Status

Use the following command to get the status of your guest system. The `-v` option gives you the tty number of your console on the IOS:

```
guest -v
```

```
gale jqp &lt;27> guest -v
gst-2 guest: Info
   guest Version 9.3.0ed, generated on 09/15/97, at 15:57:03.


              System  CPU       Address Size
Console Ord     Name  (%)       (words) (MW)  Status       Owner  Flags

 zip 0   0     gale   57              0   16  executing     HOST  M
 zip 1   1   galeg1   43      18300928   14  executing      jqp  S

Flag values:
        M (multi-user), S (single-user), F (frozen), H (halt on panic)
        B (breakpointed)

Number of CPUs available to guests:  2
Additional guest tracing is:         ENABLED
```

7.3.5 Dumping

To perform a trial dump of your guest system, use the following command:

```
guest -d
```

You do **not** have to stop the guest before dumping it. The guest system resumes execution once the dump has completed.

Keep in mind that dumps from your guest system can be quite large. You should set `DUMP_DIRECTORY=` *path* in your `guest.rc` file, where *path* is a file system with enough unused space to handle your particular requirements. For information on viewing the dump, see Appendix A.

7.3.6 Ending a session

When you have finished your guest session, execute the following command to end the session:

```
guest -q
```

This command stops the guest system (`guest -x`), releases system memory held by the guest (`guest -r`), and remounts any file systems unmounted at guest boot time.

The following is an example of the output for a guest system stop and release:

```
gale jqp &lt;40> guest -q
gst-15 guest: Info
   The guest options file:
        guest.rc
   was found in the following directory:
        /usr/guest/jqp/

gst-16 guest: Info
   Changing the current working directory to:
        /usr/guest/jqp/

gst-61 guest: Info
   The guest system (galeg1) has been successfully stopped.


gst-316 guest: Warning
   There is outstanding I/O on iopath:
        0032.0
   Waiting for I/O completion.

sys-16 guest: Info
   Device busy

gst-37 guest: Info
   The guest memory (14 MW) has been returned to host.


/etc/mfsck: Starting pass 1


/etc/mfsck: Starting pass 2
```

```
/usr_j: file system opened
/usr_j: super block fname usr_j, fpack sn1703
/src_j: file system opened
/src_j: super block fname src_j, fpack sn1703
/root_j: file system opened
/root_j: super block fname root_j, fpack sn1703
/usr_j: Phase 1 - Check Blocks and Sizes
/root_j: Phase 1 - Check Blocks and Sizes/src_j: Phase 1 - Check Blocks and Sizes
/root_j: Phase 2 - Visit Directories
/usr_j: Phase 2 - Visit Directories
/root_j: Phase 3 - Checking Directories
/root_j: Phase 4 - Checking Non-Directories and Link Counts
/root_j: Phase 5 - Verify Dynamic Information - (Ignored)
/root_j: Phase 6 - Rebuilding Dynamic Information
/root_j: file system summary
/root_j:        32768 total i-nodes (29073 free i-nodes)
/root_j:        137088 total blocks  (65395 free blocks)
/root_j:  ***** FILE SYSTEM WAS MODIFIED *****
/usr_j: Phase 3 - Checking Directories
/usr_j: Phase 4 - Checking Non-Directories and Link Counts
/usr_j: Phase 5 - Verify Dynamic Information - (Ignored)
/usr_j: Phase 6 - Rebuilding Dynamic Information
/usr_j: file system summary
/usr_j:         32768 total i-nodes (25704 free i-nodes)
/usr_j:         137088 total blocks  (39475 free blocks)
/usr_j:  ***** FILE SYSTEM WAS MODIFIED *****
/src_j: Phase 2 - Visit Directories
/src_j: Phase 3 - Checking Directories
/src_j: Phase 4 - Checking Non-Directories and Link Counts
/src_j: Phase 5 - Verify Dynamic Information - (Ignored)
/src_j: Phase 6 - Rebuilding Dynamic Information
/src_j: file system summary
/src_j:         85712 total i-nodes (45087 free i-nodes)
/src_j:         365040 total blocks  (102446 free blocks)
/src_j:  ***** FILE SYSTEM WAS MODIFIED *****
/etc/gencat: complete (9 secs.)

/gst90 is not mounted.
   Attempting to mount /dev/dsk/root_j on /gst90...done.

/gst90/usr is not mounted.
   Attempting to mount /dev/dsk/usr_j on /gst90/usr...done.

/gst90/usr/src is not mounted.
   Attempting to mount /dev/dsk/src_j on /gst90/usr/src...done.
```

Multiuser Mode Guests [8]

This chapter describes procedures for bringing up a guest in multiuser mode.

8.1 Considerations

Before bringing up a guest to multiuser mode, consider the following:

- Tapes

  Your guest's tape configuration must **exactly** match that of the host. New tape devices that are supported by the guest's system level, but not by the host, cannot be included. Tape devices are reserved in the host by device ordinal; consequently, they **must** be in the same order in the configuration. ER90 devices, though not supported on guests, should be included as ordinal place holders in the guest configuration.

- TCP/IP connection/network node name

  If you want to access your guest through a TCP/IP connection, you must have a distinct network node name and address. The next example host (`sn1703`) has a connection through an NSC A130 network adapter. The example guest uses an NSC N130 network adapter. The N130 can be shared by using a different logical path (for example, logical path 7 instead of the traditional path 5) on the guest. If you choose to use an alternate path on your existing NSC N130, be sure to have your network administrator or NSC representative update the NSC startup file. You will also need to update the `hycf` to add the adapter address for the guest. For example, with the host (`sn1703`) on logical path 5 and the guest (`sn1703a`) on logical path 7, the following lines are needed (assuming an adapter address of `0x86`):

  ```
  direct   sn1703      8605    ff00    0    16432
  direct   sn1703a     8607    ff00    0    16432
  ```

  Note that a single HIPPI or FCA-1 (FDDI) cannot be shared between host and guest for simultaneous TCP/IP connections.

  Although it is not necessary that your kernel's `uts` node name match the name by which it is known on the network, it may make it easier for a root to be used as either a host or a guest. To make a root more flexible in this sense, remove any existing `/etc/config/hostname.txt` file, and let the network's start-up scripts default to the output of the `/etc/config/makehostname` name.

8.2 Examples of configuration links

As previously mentioned, guest and host versions of selected configuration files are not necessary. However, they do facilitate the use of a single root as either a host or a guest root. With that in mind, the following examples make use of the `Guest File Link Configuration` option to prepare a root for use as either a host or a guest. You must determine which files are used at system boot time.

To create configuration links, use the following steps:

1. In the UNICOS Installation/Configuration Menu System, on the guest `root`, make the following menu selection:

   ```
   UNICOS 10.0 Installation / Configuration Menu System
   .   Configure System
   .   .   UNICOS under UNICOS (guest) Configuration
   .   .   .   Guest File Link Configuration
   ```

   For this example, the following file names are entered:

   ```
      Guest File Link Configuration

        Guest/Host Configuration Files
        ------------------------------
   E->  /etc/config/daemons
        /etc/config/interfaces
        /etc/config/ldchlist
        /etc/config/param
        /etc/config/rcoptions
        /etc/exports
        /etc/fstab
        /etc/gated.conf
        /etc/config/param
   ```

   All selected files must reside on the root file system (that is, be available in single-user mode).

2. Select and execute `Create Guest and Host versions of files`. This creates the `.host` and `.guest` files for all of the files listed in `/etc/config/guest_config`and sets links to the host version.

3. Go back to the `Activation Utility` menu, select `Activation Options`, and select `Activate host or guest versions`. Then toggle the selection to `guest` as shown in the following example:

```
.   Utilities
.   .   Activation Utility
.   .   .   Activation Options

     Activation root mount point
     Stop activation on error?                       YES
S->  Activate host or guest versions                 guest
     Activate host or guest versions.
```

4. Go back to the `Import Utility` menu, select `Import Options`, and select `Import host or guest versions`. Then toggle the selection to `guest` as shown in the following example:

```
.   Utilities
.   .   Import Utility
.   .   .   Import Options

     Import root mount point                        /
     Stop import on error?                          NO?
S-> Import host or guest versions?                  guest
     Reload default import table ...
     Activate host or guest versions.
```

5. To import your configuration using the installation tool, enter the following update screen (this example is for the `fstab` file):

```
.   Configure System
.   .   File System (fstab) Configuration
```

   You are now ready to edit the guest `fstab` information.

6. Remove references to the file systems not needed on the guest system, and activate the `fstab` configuration (this example is for the `fstab` file).

   With the `guest` attribute set in the activation utility, files listed in `/etc/config/guest_config` are linked to their `.guest` counterparts before the activation. Do not be alarmed if the activation says that it is activating to `/etc/fstab` if your intent is to update `/etc/fstab.guest`.

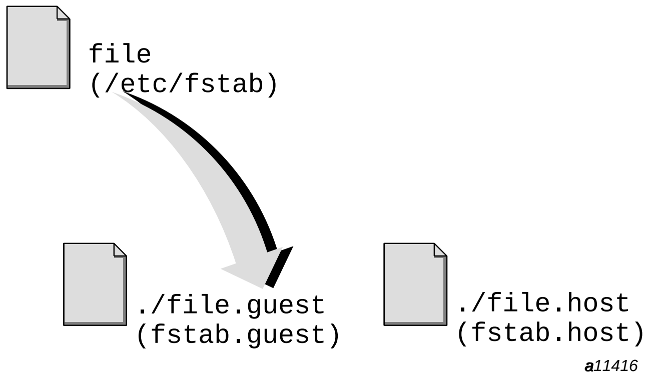


Figure 4. Example link for a system running as a guest

When a system is booted on this root, `/etc/brc.guest` determines the system type (host or guest) and appropriately sets the soft file links for all files listed in `/etc/config/guest_config`. Be sure to run `/etc/brc.guest` before you run `mfsck`(8).

Repeat step 6 for remaining files listed in `/etc/config/guest_config`.

7. Return to host import/activate mode by repeating steps 3 and 4. Choose `host` instead of `guest`.

In the example, the following changes were made in each file:

```
/etc/config/daemons.host:
    USCP  uscp            YES     /usr/lib/uscpterm /usr/lib/uscpd -d

/etc/config/daemons.guest:
    USCP  uscp            NO      /usr/lib/uscpterm /usr/lib/uscpd -d

/etc/config/interfaces.host:
    np0   /etc/hycf.ows   inet    sn1703-030       netmask 0xffffff00
    np3   /etc/hycf.hyp   inet    sn1703           netmask 0xffffff00

/etc/config/interfaces.guest:

    np1   /etc/hycf.hyp   inet    sn1703a         netmask 0xffffff00

/etc/config/ldchlist.host:
    /dev/dsk/root.a      SSD   200        48
    /dev/dsk/usr.a       SSD   175        48
    /dev/dsk/usr_spool   MEM   20         48
```

/etc/config/ldchlist.guest:

```
    /dev/dsk/root.a      SSD   200        48
    /dev/dsk/usr.a       SSD   175        48
```

/etc/config/rcoptions.host:

```
    TMPDEV='ssd_blk3'
    SDSDEV='ssd_blk0'
```

/etc/config/rcoptions.guest:

```
    TMPDEV='tmp_guest'
    SDSDEV='ssd_blk1'
```

/etc/exports.host:

```
    /sn1703
    /tmp
    /ptmp
```

/etc/exports.guest:(empty)

/etc/fstab.host:

```
/dev/dsk/root.a      /                    NC1FS  rw,CRI_RC="NO"  1      1
/dev/dsk/usr.a       /usr                 NC1FS  rw,CRI_RC="NO"  1      2
/dev/dsk/src_guest90 /usr/src             NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/usr_guest   /usr/guest           NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/root.b      /gstbld              NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/usr.b       /gstbld/usr          NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/src_guest   /gstbld/usr/src      NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/admin       /admin               NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/usr_adm     /usr/adm             NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/90_spool    /usr/spool           NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/users       /sn1703              NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/core        /core                NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/ptmp        /ptmp                NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/root.c      /90                  NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/usr.c       /90/usr              NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/src_90      /90/usr/src          NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/root.f      /90                  NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/usr.f       /90/usr              NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/src_90      /90/usr/src          NC1FS  rw,CRI_RC="YES" 1      2
/dev/dsk/root.e      /10                  NC1FS  rw,CRI_RC="NO"  1      2
/dev/dsk/usr.e       /10/usr              NC1FS  rw,CRI_RC="NO"  1      2
/dev/dsk/src_10      /10/usr/src          NC1FS  rw,CRI_RC="NO"  1      2 /proc                 /pr
```

/etc/fstab.guest(complete - removed all but root, usr, src, and proc):

```
/dev/dsk/root.a     /                 NC1F   rw,CRI_RC="NO" 1     1
```

```
/dev/dsk/usr.a      /usr              NC1FS  rw,CRI_RC="NO" 1     2
/dev/dsk/src_guest90/usr/src          NC1FS  rw,CRI_RC="YES"1     2
/proc               /proc             PROC
```

/etc/gated.conf.host:

```
     0.0.0.0 gateway zunee preference 50 hopcount 1 ;
```

/etc/gated.conf.guest:

```
     default gateway sn1703-ip preference 50 hopcount 1 ;
     cray-fddi gateway sn1703-ip preference 50 hopcount 1 ;
```

8.3 Moving to multiuser mode

After you have set up a root file system with the appropriate `.host` and `.guest` configuration links, you are now ready to boot (on that root) a multiuser mode guest. Moving to multiuser mode in a UNICOS guest system is no different than for a stand-alone UNICOS system, except for the changes that will take effect because of your configuration links.

Before booting your guest, you can run the `/mnt/etc/install/instartup.guest` script. It performs essentially the same functions as the `/etc/install/instartup` script, but can be run in multiuser mode to perform the following tasks:

- Copy the current user database (UDB) to the guest root (`/mnt/etc/udb`)
- Install the secure commands on the guest root (`/mnt`)

No options are required to copy the current UDB. To get a short help message listing the available options, enter the following command:

```
/mnt/etc/install/instartup.guest -h
```

See the steps in Chapter 7, page 23 to boot the system to single-user mode. In the guest console's window on the OWS or SWS, enter the following command:

```
/etc/init 2
```

Then proceed as you would during normal system startup.

Monitoring and Analysis [A]

This appendix describes how to analyze your environment when a guest is running or has failed.

A.1 Status

To get the status of your guest system and the tty number of your console on the IOS, use the following command:

```
guest -v
```

```
gale jqp &lt;27> guest -v
gst-2 guest: Info
   guest Version 9.3.0ed, generated on 09/15/97, at 15:57:03.


              System  CPU       Address Size
Console Ord     Name  (%)       (words) (MW)  Status       Owner  Flags

 zip 0   0     gale   57             0   16  executing     HOST  M
 zip 1   1   galeg1   43      18300928   14  executing      jqp  S

Flag values:
        M (multi-user), S (single-user), F (frozen), H (halt on panic)
        B (breakpointed)

Number of CPUs available to guests:  2
Additional guest tracing is:         ENABLED
```

A.2 CPU monitoring considerations

The `sar`(1) CPU usage output (`-u` and `-p` options) reports the amount of CPU time accumulated in concurrently running systems under a new column titled `guest`. When run in the host, `sar`(8) reports the percentage of CPU time spent in the guest under the `guest` column. When run in the guest, the `guest` column refers to host CPU time.

Across a particular time interval, the values reported in the `guest` column by the host and guest `sar`(8) command should total approximately 100%. The actual value may be a percentage point higher or lower due to rounding. Also, within a particular system, the sum of all `sar`(8) CPU usage data columns should be within a percentage point of 100.

CPUs assigned to a guest occasionally require some service provided by the host. To invoke such service, the guest CPU uses a mechanism analogous to a user process issuing a system call. The time that a CPU spends in the host kernel in servicing a request from the guest is accumulated as system time within the guest kernel. This time is recorded as `guest` time within the host kernel. The effects of this are as follows:

- System time reported by `sar` in a guest (*%sys column*) will generally be somewhat higher than in a host or stand-alone system running a similar workload. The amount of the increase depends upon the user process mix, with I/O-intensive loads showing the greatest increase.
- System time reported by `timex`(1) generally will be higher in a guest. The size of the increase depends upon the number of host requests the guest kernel must issue during the handling of each user system call issued during the timed interval.
- Wall-clock time (reported as *real* by `timex`) for a particular process may be greater in a guest than in a host or stand-alone system. In general, the amount of increase is proportional to the amount of I/O performed by the process.
- User time values are independent of the kernel's mode of operation (host versus guest). Not only are user CPU times reported by `sar` independent of the kernel mode but user CPU time associated with a particular process also is independent. For example, `timex` reports the same user time for a process when run in a guest as it would for the same process run in the host. User time may vary due to bank conflicts.
- It is possible to monitor the host request activity (host calls issued by an active guest kernel) using a new `sar` option, `-g`. For an explanation of the sample information that follows, refer to the `sar`(1) man page.

```
sn1703% sar -g 10 1000

sn1703a 7.0.0 hst.305 CRAY Y-MP    04/20/94

17:25:59 host call        %time   calls/s   avetime     maxtime   mintime
17:26:10 I/O interrupt        3.4       2.5     115.8     28960.9      26.6
         low speed I/O        2.5       2.9      72.9     16193.2      31.3
         clk interrupt        2.2       1.0     185.3      1466.0      63.5
            return CPU         na       0.4        na          na        na
               SSD I/O        0.1       0.1      52.5      5359.2      31.3
         user exchange       86.6     402.0      18.5    316646.0      12.4
           get cluster        2.5       5.7      38.1      3906.7      31.8
           put cluster        2.7       5.7      41.1      3909.7      32.0

17:26:21 I/O interrupt        7.2       2.7      80.4     28960.9      26.6
         low speed I/O        3.6       1.5      74.2     16193.2      31.3
            return CPU         na       5.5        na          na        na
               SSD I/O        3.3       1.9      52.1      5359.2      31.3
         user exchange       68.5     112.5      18.5    316646.0      12.4
           get cluster        9.1       7.3      37.9      3906.7      31.8
           put cluster        8.4       6.7      37.9      3909.7      32.0
sn1703a sn1703 7.0.0 hst.305 CRAY Y-MP    04/20/94
```

A.3 Dumping

To dump your guest system, use the following command:

```
guest -d
```

It is **not necessary** to stop the guest before dumping it.

At any time during the running of your guest, you may dump the system by using the `guest -d` command. If the guest system is not panicked, the host kernel will set the internal freeze (inhibit) flag for the guest system while the dump is in progress. This prevents CPUs from exchanging to user processes in the guest, thus, preserving the integrity of the dump. The freeze flag is cleared when the dump completes successfully.

As noted in the following dump output example, the `guest`(1) command informs the user of how the dump is proceeding and where the dump and binary files will be placed. A guest dump directory contains `crash`(8) and UNICOS binaries for all systems occupying mainframe memory. A *_guest* name extension is added to the guest-related binaries in the dump directory. Guest system dumps can be much larger than host dumps (up to twice the size), so you may need to increase your dump partition size for mainframe dumps taken from the OWS or SWS. You should set `DUMP_DIRECTORY=`*path* in your `guest.rc` file, where *path* is a file system with enough unused space to handle your particular requirements.

In the dump directory, you start the `crash`(8) command as follows:

```
./crash dump unicos
```

All `crash`(8) commands entered at this point are relative and pertain only to the host’s system memory. You may view the guest portion of the dump by entering the `guest`(1) command at the `crash>` prompt. If `crash` finds the appropriate binaries (denoted by guest system name) in the current directory it will ask you if you want to fork a new crash to examine the dump. Unless the system levels are virtually identical, an affirmative response is usually appropriate.

After forking the new crash or allowing the current crash to change its memory bias, all `crash`(8) commands will be relative to the guest’s system memory. All `crash` commands (except for some UNICOS under UNICOS specific commands) are available when analyzing a guest dump. For more information, see the `crash`(8) man page.

An example of the `-d` option of the `guest`(1) command follows:

```
gale jqp <34> guest -d
Enter a dump comment (up to 80 characters):
tpconfig process hung in guest
gst-23 guest: Info
   Guest dump files will be located below the following directory:
        ./dumps/04050534

Dump segment address:    030412000   cumulative words:    012110000
gst-25 guest: Info
   Successfully completed a guest dump of the system named: gale

Dump segment address:    077610700   cumulative words:    022703000
gst-25 guest: Info

   Successfully completed a guest dump of the system named: galeg1

gale jqp <35> cd ./dumps/04050534
/usr/guest/jqp/dumps/04050534

gale 04050534 <36> ls
crash          crash_galeg1   dump           unicos         unicos_galeg1

gale 04050534 <37> ./crash dump unicos
> guest
Use ./crash_galeg1 (y|n)? > y
Running ./crash_galeg1 -s -g galeg1 dump ./unicos_galeg1
> ut
+          0000000000000000000010 0000000000000000000000 0000000000000000000000
host2->    0030701403117621106355 0000000000000000000001 0000000000000000000000
+          0000000000000000000003 0000000000000000000000 0000000000000000000000
host1->    0030701403117621106160 0000000000000000000000 0000000000000000000010
host->     0030701403117621106025 0000000000500000000000 0000000000000000000000
+          0000000000000000000010 0000000000000000000000 0000000000000000000000
->host2    0030701403117602533114 0000000000000000000001 0000000000000000000000
+          0000000000000000000003 0000000000000000000000 0000000000000000000000
->host1    0030701403117602532722 0000000000000000000000 0000000000000000000010
->host     0030701403117602532565 0000000000500000000000 0000701403117621511220
clockown   0030701403117602531633 0000000000000000000004 0000000000000000000000
+          0000000000000000000017 0000000000000000000010 0000000000000000000000
routcp5    0030701403117602530357 0000000000000000000010 0000000000000000000003
+          0000000000000000000003 0000000000000000000010 0000000000000000000000
```

```
routcp4    0030701403117602530007 0000000000000000000000 0000000000000000000010
+          0000000000000000000010 0000000000000000000004 0000000000000000001040
host2->    0030701403117602526717 0000000000000000000000 0000000000000000000000
+          0000000000000000000003 0000000000000000000010 0000000000000000000010
host1->    0030701403117602526543 0000000000000000000000 0000000000000000000010
host->     0030701403117602526410 0000000000200000000000 0000000000000000000000
+          0000000000000000000010 0000000000000000000004 0000000000000000000000
->host2    0030701403117602141211 0000000000000000000000 0000000000000000000000
+          0000000000000000000003 0000000000000000000010 0000000000000000000010
->host1    0030701403117602141014 0000000000000000000000 0000000000000000000010
->host     0030701403117602140665 0000000000200000001040 0000000000000000000000
swtchc     0030701403117602137124 0000022620300007571000 0645443306240000000000
swtchg1    0030701403117602134176 0000000000000000000010 0000000000000000000001
swtchd     0030701403117602130560 0000022620300007571000 0645443306240000000000
+          0000000000000000000017 0000000000000000000010 0000000000000000000000
routcp5    0030701403117602127557 0000000000000000000010 0000000000000000000003
+          0000000000000000000003 0000000000000000000010 0000000000000000000000
routcp4    0030701403117602127203 0000000000000000000000 0000000000000000000010


> p
SLT ST  PID  PPID  PGRP   UID  EUID  PRI CPU   EVENT  NAME     FLAGS
  0 s     0     0     0     0     0    0   - 00223731 sched    Ld Sys
  1 s     1     0     1     0     0   39   - 00205552 init     Ld rwt
  2 r     2     0     0     0     0  999   0     -    idle     Ld Sys Idl conn
  3 r     3     0     0     0     0  999   1     -    idle     Ld Sys Idl conn
  4 r     4     0     0     0     0  999   2     -    idle     Ld Sys Idl conn
  5 r     5     0     0     0     0  999   3     -    idle     Ld Sys Idl conn
  6 s  1127     1  1127   526   526   39   - 00205552 csh      Ld Nocore rwt
  6                                                            oldmask
  7 s  1129     1  1129     0     0   28   - 01013567 getty    Ld
  8 s  7260  7252  7260     0     0   39   - 00205552 csh      Ld rwt oldmask
  9 s   380     1   380     0     0   26   - 02322001 slogdemo Ld Plock
 10 s  1128     1  1128     0     0   28   - 01013521 getty    Ld
 11 s   621     1   621     0     0   26   - 00730374 cron     Ld
 12 s   608     1   301     0     0   39   - 00205552 shrdaemo Ld rwt oldmask
 13 r   634     1   634     0     0   39   -     -    fsdaemon Ld rwt oldmask
 14 s   626     1   626     0     0   26   - 01232235 msgdaemo Ld
 15 s   777     1   777     0     0   26   - 01232235 inetd    Ld
 17 s   792     1   792     0     0   26   - 01232235 lpd      Ld
 18 s   782     1   301     0     0   26   - 05264014 sendmail Ld
 19 s   984     1   984     0     0   40   - 07300071 tpdaemon Ld
 20 s   959     1   301     0     0   26   - 01232235 errdemon Ld
 21 s   799     1   301     0     0   26   - 01232235 snmpd    Ld
 22 s   812     1   812     0     0   26   - 01232235 portmap  Ld
 23 s   850     1   850     0     0   26   - 01232235 ypserv   Ld
 24 s   968     1   968     0     0   26   - 01232235 syslogd  Ld
 26 r   975     1   975     0     0   26   -     -    crayperf Ld
 27 r  1135   984   984     0     0   39   -     -    slnet    Ld rwt oldmask
 28 s   859     1   859     0     0   26   - 01232235 ypbind   Ld
 29 s  1304     1  1304     0     0   26   - 05463020 nfsd     Ld
 30 s  1092  1014  1014     0     0   39   - 00205552 qfdaemon Ld rwt
 31 s  1088  1014  1014     0     0   26   - 05325614 netdaemo Ld
 32 s  1305  1304  1304     0     0   26   - 05463020 nfsd     Ld
 33 s  1013     1  1013     0     0   26   - 00671340 logdaemo Ld
 34 s  1014     1  1014     0     0   26   - 00672324 nqsdaemo Ld
 35 s  1117     1  1113     0     0   26   - 05331414 rtidaemo Ld
 36 s  1144   984   984     0     0   39   - 00205552 avrproc  Ld rwt
 37 s  1306  1304  1304     0     0   26   - 05463020 nfsd     Ld
 38 s  1307  1304  1304     0     0   26   - 05463020 nfsd     Ld
 39 s  1313  1312  1312     0     0   26   - 05466620 cnfsd    Ld
 40 s  1312     1  1312     0     0   26   - 05466620 cnfsd    Ld
 41 s  1314  1312  1312     0     0   26   - 05466620 cnfsd    Ld
 42 s  1315  1312  1312     0     0   26   - 05466620 cnfsd    Ld

 44 s  1324     1  1324     0     0   26   - 01232235 mountd   Ld
 45 s  1329     1  1329     0     0   26   - 02246673 biod     Ld
 46 s  1330     1  1330     0     0   26   - 02246673 biod     Ld
 47 s  1331     1  1331     0     0   26   - 02246673 biod     Ld
 48 s  1332     1  1332     0     0   26   - 02246673 biod     Ld
 49 s  7250     1  7250     0     0   26   - 01232235 automoun Ld
 51 s  7271  7260  7271     0     0   26   - 00704723 tpconfig Ld
 53 s  7248     1  7248     0     0   26   - 01232235 automoun Ld
 54 s  7252  1127  7252   526     0   30   - 00253176 zup      Ld Nocore rwt
 55 s  7255     1  7255     0     0   26   - 01232235 automoun Ld
 56 s  7253     1  7253     0     0   26   - 01232235 automoun Ld
 57 s  7257     1  7257     0     0   26   - 01232235 automoun Ld
 58 s  7259     1  7259     0     0   26   - 01232235 automoun Ld
> stack 51
u_save[0].B01=01735147, u_save[0].B02=010122231, &u_stack[0]=010122057
```

```
KERNEL STACK TRACE for proc slot 51:

 routine name          entry point  return addr  line #   arguments

 swtch                 01737312c    01734626b    *** arg list ptr out of bounds
 sleep                 01734353c    01400026b    315      0000000000000000704723
                                                          0000000000000000004032
 nc1pread              01377655c    01741022a    162      0000000000000000602250
                                                          0000000000000000000000
 rdwr                  01740534c    01740517a    63       0000000000000000000001
 read                  01740506c    01606140a    *** arg list ptr out of bounds
 umain                 01670606c    01607612b
> stack 19
u_save[0].B01=01735147, u_save[0].B02=020220305, &u_stack[0]=020220057
KERNEL STACK TRACE for proc slot 19:

 routine name          entry point  return addr  line #   arguments

 swtch                 01737312c    01734626b    *** arg list ptr out of bounds
 sleep                 01734353c    01536700d    2866     0000000000000007300071
                                                          0000000000000000001724
 bmxsleep              01536641c    01536332d    2693     0000000000000007300071
                                                          0000000000000000001724
 bmxaux                01536203c    01527513b    604      0000000000000007300071
                                                          0000000000000000000005
                                                          0000000000000000000020
                                                          0000000000000000000001
 bmxclose              01526767c    01407254a    693      0000000000000000600112
                                                          0000000000000000000001
 nc1closei             01407050c    01645755a    110      0000000000000000600112
                                                          0000000000000000000001
                                                          0000000000000000000001
                                                          0000000000000000000000
                                                          0000000000000000511011
 closef                01645574c    01745033c    1238     0000000000000000511011
                                                          0000000000000000000013
 close                 01745004c    01606140a    *** arg list ptr out of bounds
 umain                 01670606c    01607612b
> q
Back in host crash.
> q
```

Troubleshooting [B]

This appendix describes situations that you may encounter and the steps you can take to resolve them.

B.1 `guest` command fails

The `guest`(1) command can fail because it cannot obtain the desired guest memory. If this occurs, do the following:

1. Increase the memory retry count (`MEMORY_RETRIES=#`) in your `guest.rc` file and run the command again.
2. If retrying is unsuccessful, suspend longer processes until guest memory can be allocated.

B.2 `cannot open root device` message

If the guest system panics immediately with a `cannot open root device` message, do the following:

1. Ensure that the root file system specified in the guest parameter file is not mounted on the host.
2. Check the `guest`(1) command output to ensure that the file system was unmounted without error (if the logical device was specified in the user's `guest.rc` file).

B.3 NQS does not start on the guest

If the Network Queuing System (NQS) does not start on the guest, check the NQS configuration for the guest to ensure that a unique machine ID (`mid`) is specified for the guest's TCP host name. The `/etc/config/nqs_config` file contains the machine IDs.

B.4 Guest does not recognize tape mounts

If the guest does not recognize tape mounts, even though the drive appears to be configured up on the guest, ensure that the host and guest configurations are identical. Device ordinals for each drive must be the same. Check the output of the `tpdev`(8) command on both the host and guest system. If the device ordinals do not match, reorder the devices in the guest's `/etc/config/tapeconfig` file, and restart both tape daemons.

B.5 Kernel decompression loops forever

If the kernel decompression step of the guest startup seems to run forever, make sure that the `kcompress`(8) version (`KCOMPRESS` entry in your `guest.rc` file) matches the guest kernel being booted. It is normal for this step to take several (15 or more) seconds.

guest Messages [C]

Error messages that can be issued by the guest(1) command are listed in the online message catalog. The group code for this product is guest. Each message is listed, along with an extended explanation that describes the error in greater detail and suggests actions for solving the problem.

Index