CONVEX SPU OS V6.3 Release Notice

Document No. 760-004830-004

September 1993

CONVEX Press
Richardson, Texas USA

CONVEX
SPU OS V6.3
Release Notice

Document No. 760-004830-004

Printed in the United States of America

Contents

1. Introduction

This document describes V6.3 of the CONVEX SPU OS operating system. This document is intended to enhance and clarify the existing permanent documentation for this product with information that is up-to-the-minute, or was developed too late for inclusion in the permanent documentation. Always refer to this release notice before reporting questions or problems with CONVEX SPU OS. Your questions may be answered here. Fixes and workarounds are listed here that may save you time in rediscovering known problems.

CONVEX SPU OS 6.3 supports 3 different kinds of CONVEX Service Processor Units (SPU): The "sp2" (part number 410-001200-200); the "sp4" (part number 410-002223-200); and the "sp5" (part number 410-002223-200). From the standpoint of this document, the sp2, sp4, and sp5 are functionally identical.

The remaining sections in this document describe the contents of this release.

- Section 2 describes the contents of the distribution.
- Section 3 contains notes and warnings about the use of the software.
- Section 4 contains enhancements to the previous functionality.
- Section 5 describes fixes for previously reported problems.
- Section 6 describes known software problems.
- Section 7 contains known documentation problems.
- Section 8 contains new documentation.
- Appendix A contains instructions for installing this release on a CONVEX Service Processor Unit (sp2, sp4, or sp5).
- Appendix B contains instructions for setting the default timezone.
- Appendix C contains new and replacement man pages for this release.

CONVEX SPU OS is the sole operating system for the CONVEX SPU and is derived from AT&T UNIX Version 7. CONVEX SPU OS is used as the off-line diagnostic environment for all diagnostic test programs and utilities.

2. Contents of this distribution

The distribution package for this release of CONVEX SPU OS consists of this document and a set of distribution media for the software. Only one type of distribution media are used for CONVEX SPU OS, using the DC600A data cartridge. For both sp2 and sp5, QIC (yellow label) is the only distribution media format.

Table 1: CONVEX SPU OS Software Release Package

Item	Quantity	Type	Part Number	Description	Format
1	1	QIC	760-001215-209	CONVEX SPU OS V6.3	Installsw/Boot Image

Table 2: CONVEX SPU OS Documentation Release Package

Item	Quantity	Type	Part Number	Description
1	1	Release Notice	760-004830-004	CONVEX SPU OS V6.3 Release Notice
1	1	Manual	760-000530-000	CONVEX SPU OS Utilities Manual, 2nd Edition

3. Notes and Warnings

This section contains general useful information or words of caution about the product.

- This release supersedes CONVEX SPU OS V6.0, V6.1, and V6.2.
- This release only supports C200, C3200, C3400, and P5500 Series systems.
- `/etc/timezone` is no longer supported as the method of setting the default timezone. Use the new `zic` utility instead.
- A new message that identifies the manufacturer of the root disk as been added to the SPU OS boot process. For example, for a Seagate ST11200N, the message would appear in the boot sequence as:

```
available memory = ####### (#### Kbytes)
Disk Vendor Info(32): SEAGATE ST11200N
SPU root file system check in progress...
```

- For purposes of compatibility with older software, the `tzsetup` utility must be run after the default local timezone has been set to ensure that the correct date and time are displayed by all utilities.

4. Enhancements

4.1 Files

- The maximum number of open files per process has been increased from 20 to 40.
- The `/etc/zoneinfo` database has been added to support the `date`, `zic`, and `zdump` utilities, as well as other utilities that use the `ctime(3)` library call.

4.2 Utilities

- The `date` utility has been modified to use the `/etc/zoneinfo` database.
- The Zone Info Compiler (`zic`) has been added to create the `/etc/zoneinfo` database from the files found in `/etc/zoneinfo/data`.

- The `zdump` utility has been added to display the current time from the perspecitive of non-default timezones.
- The `tzsetup` utility has been added to make older software compatible with the new date and time support.

4.3 SPU OS

- The maximum number of open files per process has been increased from 20 to 40.
- SPU OS has been modified to more intelligently deal with, and take fullest advantage of, the various disk drives that can be installed on C200, C3200, C3400, and P5500 Series systems. This includes the new 1 GB SPU disk drives that will soon begin shipping in 3Q93.
- SPU OS has been modified to increase the size of the buffer allocated to scan rings on C3400 systems to allow both versions of C3400 CPUs to be installed on a single system.

5. Fixes

The following bug fixes have been made to this release:

- The `restore` utility in the 6.01 version of CONVEX SPU OS did not rebuild file systems properly before restoring a backup tape's contents back onto the SPU disk. This problem would cause the fsck utility to hang when rebooting the SPU after such a restore was done. This has been fixed in the 6.3 version of CONVEX SPU OS. Please see the "Utilities" paragraph of the "Known Software Problems" section for an erroneous message that restore may produce.
- Various bugs reported against the `date` command have been fixed. This includes fixing daylight savings time (DST) rules to be in line with ConvexOS 11.0.
- `/etc/backup` has been modified to do "`/bin/mt rewind`" prior to starting the backup to ensure that the cartridge tape is correctly positioned prior to actually starting the backup.
- There was a problem with swap space management that sometimes would cause the fork of a child process to fail (e.g. when you "bang" a shell from `cpu4241`). This problem has existed at least as far back as SPU OS V5.X). It has now been fixed.
- There was a problem with date/time management that was introduced with the date/time bug fixes in SPU OS V6.2. This problem has been fixed with the creation of the `tzsetup` utility.

6. Known Software Problems and Limitations

As of the time of the preparation of this release notice, this section contains the known problems and limitations with the CONVEX SPU OS software. Please refer to this section prior to reporting a problem in order to ensure that it has not been previously reported. Serious problems include workarounds if they are known.

6.1 Operating System

The following limitation with `csh` script execution has been identified:

- Invocation of csh scripts that automatically use the csh to run the script through the mechanism of making the first line of the script '`#!/bin/csh`' is not supported in this release.

6.2 Utilities

The following problems and limitations have been identified with various SPU OS utilities:

- The `restore` utility may produce the following error message:

  ```
  Attempt to mount <directory> on <device> FAILED:
  Mount device busy
  ```

  This is NOT an error condition, just an erroneous error message. It can safely be ignored.

- `Tar` does not support the u and r options.
- The "-r" option of `installsw(8)` does not work properly if the tape is a QIC version of the CONVEX SPU OS release tape.

7. Known Documentation Problems

The following problem with SPU OS documentation has been identified:

- The `date` man page in the *SPU OS Utilities Manual* is obsolete, and should be replaced with the man page found in Appendix C of this document.

8. New Documentation

The following new documentation has been associated with this release:

- Replacement man pages are included in this document for the following utilities:
  - `backup(8)`
  - `ctime(3)`
  - `date(1)`
  - `gettimeofday(3)`
  - `test(1)`
  - `tzset(3)`
- Man pages for the following new utilities may be found at the end of this document:
  - `tzfile(5)`
  - `tzsetup(8)`
  - `zdump(8)`
  - `zic(8)`

I. Installing CONVEX SPU OS V6.3

CONVEX SPU OS 6.3 supports 3 different kinds of CONVEX Service Processor Units (SPU): The "sp2" (part number 410-001200-200); the "sp4" (part number 410-002223-200); and the "sp5" (part number 410-002223-200). From the standpoint of this document, the sp2, sp4, and sp5 are functionally identical.

Before performing the installation of this product, the following information should be examined:

- This release supersedes CONVEX SPU OS V6.0, V6.1, and V6.2.
- This release only supports C200, C3200, C3400, P5500 Series systems.

The remainder of Appendix A describes procedures for installing CONVEX SPU OS V6.3. The first describes installing the release on a system on which a working version of CONVEX SPU OS already exists, and the second describes installing the release on a new, or non-bootable, disk drive. Note: IOmega drives are **not** supported root devices.

The procedures are described in the following sections:

- Installation on a Bootable SPU Disk Drive
- Installation on a Non-bootable SPU Disk Drive

I.1 Installation on a Bootable SPU Disk Drive

Always use this procedure to install CONVEX SPU OS V6.3, except when installing the release on non-bootable hardware or when the previously installed version of CONVEX SPU OS will not boot and execute. If either of those two conditions exists, use the procedure for "Installation on a Non-bootable SPU Disk Drive".

1. If CONVEX SPU OS is already booted, go to step 5.
2. Place the front panel key switch in the *local* position and depress the system reset button. Note that when the key switch is in this position, it is not possible to access the remote console.
3. The soft front panel menu will be displayed. Change the mode to diagnostics and continue the boot process by entering the following commands at the (fp)> prompt:

   ```
   (fp)> set mode=diagnostic

   (fp)> boot
   ```

4. The CONVEX SPU OS bootstrap routine will then prompt with:

   ```
   SPU OS boot

   :<CR>
   ```

Enter a carriage return <CR> in response to the prompt. CONVEX SPU OS will now boot and prompt with (spu)> when boot is complete.

NOTE: A file system check is performed during the boot procedure. If errors are detected, they will be corrected if possible. If it is not possible to automatically correct the errors, you will be requested to execute /etc/fsck manually to correct them before proceeding.

5. If ConvexOS is booted, perform a `shutdown(8)` before proceeding.

6. If steps 2-4 were skipped, execute `/etc/fsck` manually and correct all errors before proceeding:

   ```
   (spu)> /etc/fsck -y
   ```

   If errors are detected in the root file system, reboot CONVEX SPU OS as follows:

   ```
   (spu)> /etc/reboot -n

   (fp)> boot

   SPU OS boot

   :<CR>
   ```

   Enter a carriage return <CR> in response to the ":" prompt.

7. Obtain a scratch cartridge tape and make sure that the writeprotect indicator is not set to the "Safe" position.

8. Place the scratch tape in the cartridge tape unit. Backup the SPU disk by entering:

   ```
   (spu)> /etc/backup
   ```

   This process will take 20-30 minutes depending upon the number of files on the SPU disk and on the type of tape drive being used.

9. After the backup tape is complete, remove it from the cartridge tape unit and move the write-protect indicator to the "Safe" position. Label the tape appropriately and put it in a place for safekeeping.

10. Place the CONVEX SPU OS V6.3 install tape (P/N 760-001215-209) in the cartridge tape unit.

11. Install the new release by entering:

    ```
    (spu)> /etc/installsw -i
    ```

    The CONVEX SPU OS software will be read from the tape and installed on the Winchester. Unlike past releases of SPU OS, you will no longer be prompted to enter a default timezone. See the man pages in Appendix 2 for information on how to set the default timezone.

12. You will be prompted to place the keyswitch in *local* mode, after which the SPU will return to the front panel. At the soft front panel prompt, reboot the SPU by entering:

    ```
    (fp)> boot
    ```

13. If the CONVEX SPU OS bootstrap routine prompts with:

    ```
    SPU OS boot...

    :<CR>
    ```

    enter a carriage return <CR> in response to the prompt. CONVEX SPU OS will now boot and prompt with `(spu)>` when boot is complete.

14. Verify that the `tar` of the appropriate files completed without error by examining the file */tmp/installsw.tar* and looking for messages that might indicate an error occurred, such as:

    ```
    tar: tape blocksize error
    ```

    or

    ```
    tar: directory checksum error
    ```

    If it appears that the `tar` failed, determine the cause of the failure and re-execute the `/etc/installsw` command. If the `tar` was successful, remove the log file:

    ```
    (spu)> rm /tmp/installsw.tar
    ```

15. After the installation is complete, remove the tape from the cartridge tape unit.

16. If the desired mode of operation is diagnostic mode, then this step may be skipped. Otherwise, return to the soft front panel via the `/etc/reboot` command:

    ```
    (spu)> /etc/reboot
    ```

    Change the mode of operation setting to the desired mode. Use the soft front panel `help` command if you need assistance.

    ```
    (fp)> set mode=<desired mode>
    ```

    Place the front panel key switch in the *secure* position and enter the boot command to reboot the system:

    ```
    (fp)> boot
    ```

17. This completes the installation of CONVEX SPU OS V6.3.

I.2 Installation on a Non-bootable SPU Disk Drive

This procedure should be used for installing CONVEX SPU OS V6.3 onto a new Winchester disk or onto a system on which the previously installed version of CONVEX SPU OS will not boot and execute. Note that this procedure only installs the root image, and any files outside of the SPU OS root partition must be loaded from a backup tape or from other SPU product release tapes. See the release notices for these other SPU products for more information on installation dependencies, procedures, etc.

1. Place the front panel key switch in the *local* position and depress the system reset button. Note that when the key switch is in this position, it is not possible to access the remote console.

2. Place the appropriate CONVEX SPU OS Installsw/Boot Image tape (V6.3 760-001215-209) in the cartridge tape drive.

3. The soft front panel menu will be displayed. Change the mode to diagnostics and continue the boot process by entering the following commands at the `(fp)>` prompt:

    ```
    (fp)> set mode=diagnostic
    (fp)> set boot=tape
    (fp)> boot
    ```

4. The SPU will boot from the cartridge tape and the disk/tape utility (spu2000) will display its menu. Either uppercase or lowercase characters may be used for all responses.

```
SPU cartridge tape boot...

Loading copy 0...

SPU Disk/Tape Diagnostic Utility

  (O) for OS Root Restore

  (D) for Disk/Tape Utility

  (S) for SPU Hardware Utility

  (R) for Reboot SPU

Enter utility to execute -
```

5. The SPU Winchester disk can be formatted at this time if needed. If the SPU Winchester disk is formatted properly, then skip this step.

   To format the SPU Winchester, enter the following sequence of commands.

```
SPU Disk/Tape Diagnostic Utility

  (O) for OS Root Restore

  (D) for Disk/Tape Utility

  (S) for SPU Hardware Utility

  (R) for Reboot SPU

Enter utility to execute - d


SPU Format Disk/Tape Utility

  (D) for Disk (SPU Winchester)

  (T) for Tape (SPU cartridge)

  (I) for IOmega (SPU removable disk)

  (E) for Exit test

Enter controller type/function - d

Format desired using standard defaults and no
prompts [yn] (y)? n
```

   This may then be followed by an informational message that identifies the disk drive that is installed on the SPU. Note that the exact contents of this message will vary depending on the disk drive installed.

```
  SCSI Inquiry: Vendor Info(7): SEAGATE ST11200N
```

   Now tell spu2000 to format the disk:

```
Format/test, Debug, or Abort operation [F,D,A]? f

Enter the following:

Run maintenance track test? ------[yn] (n) -> <CR>
```

```
Run format test? -----------------[yn] (n) -> y
Run write test -------------------[yn] (n) -> n
Run read test? -------------------[yn] (n) -> n
Run bad block f------------------[yn] (n) -> n
Run random read test? ------------[yn] (n) -> n
Run seek test? -------------------[yn] (n) -> n
LOOP ON TESTS? -------------------[yn] (n) -> <CR>
MAX NUMBER OF ERRORS ---------(1) -> <CR>
 Are all inputs correct? [yn] -> y
```

NOTE: The bad block fix test is to be run if user has specific locations on the disk that should be considered "bad". These could be relative block locations or they could be entries from the manufacturer's defect list supplied with the disk. The use of this routine is not needed unless defects are to be input by the user.

The format operation will take from 10-30 minutes to complete depending on the size of the SPU disk. It is strongly recommended that the write and read test not be run, since the write test takes approximately 10 hours, and the read test takes approximately 7.

Answer the follow-up prompt with an "a" for abort. This will return the utility to the top level menu.

```
 Format, Debug or Abort operation [F,D,A]? a
```

6. The CONVEX SPU OS root image can be restored once the format of the SPU Winchester is complete; this image consists of the boot tracks and the OS root file system. The command sequence to enter is as follows. Note that the following output will differ slightly depending on the type of machine and the type of tape drive being used.

```
SPU Disk/Tape Diagnostic Utility $Revision 2.1 $
 (O) for OS Root Restore
 (D) for Disk/Tape Utility
 (S) for SPU Hardware Utility
 (R) for Reboot SPU
Enter utility to execute - o

SPU OS Root Partition Restore
 reading bad block table...0
  Attempting sector: 0 Successful.
 reading root date code...0
  Attempting sector: 0 Successful.

SPU OS root size = 3994 blocks. Backed up....
```

```
Restore root onto disk or IOmega? [di] (d) d
```

This may then be followed by an informational message that identifies the disk drive that is installed on the SPU. Note that the exact contents of this message will vary depending on the disk drive installed.

```
  SCSI Inquiry: Vendor Info(7): SEAGATE ST11200N
```

Now install the root partition:

```
Recover the root at this time? [yn] (n) y

Recover copy 0 or 1? (01) [0] <CR>
```

The restore should take about 8 minutes. The data written to the Winchester disk is verified as it is written; any data mismatches indicate either bad hardware or a badly-formatted disk. However, this procedure does not verify that data read from the tape is not corrupt, so two copies of the root image are provided; if copy 0 is bad, repeat the above procedure and answer the last prompt for recover copy with a 1.

7. At this time, answer the top level menu with an "r" to reboot the SPU, and this will return the SPU to the front panel monitor.

```
SPU Disk/Tape Diagnostic Utility

 (O) for OS Root Restore

 (D) for Disk/Tape Utility

 (S) for SPU Hardware Utility

 (R) for Reboot SPU

Enter utility to execute - r
```

The soft front panel menu will be displayed. Change the boot device to disk and continue the boot process by entering the following commands at the `(fp)>` prompt:

```
(fp)> set boot=disk

(fp)> boot
```

8. The CONVEX SPU OS bootstrap routine will then prompt with:

```
SPU OS boot

:<CR>
```

Enter a carriage return <CR> in response to the prompt. CONVEX SPU OS will now boot and prompt with `(spu)>` when boot is complete.

9. If this installation is on a non-bootable SPU disk drive, go to step 10. If this installation is on a bootable disk whose previously mountable file system(s) are intact, perform a file system check on all file systems and then mount them:

```
(spu)> /etc/fsck

(spu)> mount -a
```

Once the file system is mounted, proceed to step 11.

10. If this installation is on a non-bootable SPU disk drive, then the mountable file systems (*/mnt*, */hw*, and */tmp* on the sp2/sp4, */mnt*, */hw*,

/tmp, */sst*, and */scratch* on sp5), will need to be created at this time. Create these file systems by entering:

```
(spu)> /etc/restore
```

This step should take about 10 minutes to complete. Note that if no backup tape is installed, a few error messages from attempted tape access will be generated and can be ignored.

NOTE: If no backup tape was installed for the `/etc/restore` operation, there will now be empty mountable file system(s). You will need to follow the procedures for installing the appropriate versions of other SPU-resident software such as System Diagnostics, Diagnostics Database, and Convex OS following the completion of installation of CONVEX SPU OS.

11. Set up the I/O configuration file for your system:

    ```
    (spu)> cp ioconfig.skel ioconfig
    ```

    Display the *ioconfig* file using `more` or `less`. If this file does not agree with the configuration of the system (refer to the system configuration in the site log for details), then use `vi` or `xed` to edit the file appropriately. If you do not know how to do this, contact the CONVEX Technical Assistance Center (TAC).

12. This completes the installation of CONVEX SPU OS V6.3.

J. Setting a default timezone

In previous releases of SPU OS, the default timezone was set by placing the timezone into a file called `/etc/timezone`. Then, each time that the SPU was booted, the SPU OS kernel was informed of the default timezone with the following command (executed in `/.profile` as a part of the boot process):

```
/bin/date -z '/bin/cat /etc/timezone'
```

Over time, however, inconsistencies have arisen between SPU OS and ConvexOS and how and when they calculated daylight savings time (DST) transitions. This occasionally caused the SPU and the C Series mainframe to report different times because of different DST rules that have evolved. SPU OS has now been modified to utilize a new utility, the Zone Info Compiler (`zic`) and the database that it creates, `/etc/zoneinfo`. This utility and database will now permit the actual DST rules to continue to evolve because the timezone information is stored in the database rather than actually being compiled into the application. This identical mechanism will be used in ConvexOS, beginning with ConvexOS V11.0.

The following is an example of how to set a default timezone. In this example, we will assume that we are only interested in being able to understand timezones in North America and Europe, and we will set our default timezone to be US Central Standard Time.

1. Change to the directory where the timezone rule files are stored:

   ```
   (spu)> cd /etc/zoneinfo/data
   ```

2. Compile the desired rule sets using the Zone Info Compiler:

   ```
   (spu)> zic northamerica europe
   ```

   This creates a number of subdirectories and sub-subdirectories under `/etc/zoneinfo` that contain the "compiled" timezone information. The following rule set files are provided:

   - africa
   - antarctica
   - asia
   - australasia
   - europe
   - northamerica
   - pacificnew
   - southamerica

   View these files to determine the actual timezones that are defined within each, as well as the paths to each of the "compiled" database files.

3. Set up the default timezone. In our case, it will be US Central Standard Time (CST).

   ```
   (spu)> zic -l NorthAmerica/US/Central
   ```

   This creates a "hard" link from the file `/etc/zoneinfo/localtime` to the file `/etc/zoneinfo/NorthAmerica/US/Central`, which

was created when the Zone Info Compiler was run on the `northamerica` rule set.

4. The default timezone is now set, and date will correctly report the local date and time using the rules from the file linked to by `/etc/zoneinfo/localtime`.

5. Now run the utility `tzsetup`. It causes the correct timezone and daylight savings time information (as indicated by `/etc/zoneinfo/localtime`) to be updated in the SPU OS kernel. This will permit some tools that do not use the new date/time utilities to report the correct date and time when they are executed.

   ```
   (spu)> tzsetup
   ```

   `Tzsetup` is also called from `/.profile` when SPU OS boots, so there is no need to run this utility by hand after rebooting SPU OS unless the default timezone is changed again.

6. To check the time of another timezone (that has been compiled with `zic`), use the `zdump` command. For example, using the previous example, to determine the current time in Newfoundland, one would enter:

   ```
   (spu)> zdump NorthAmerica/Canada/Newfoundland
   ```

7. For more information, read the man pages for `date`, `zic`, and `zdump` in Appendix C.

K. SPU OS Man Pages

This section includes new and replacement man pages for SPU OS utilities. The following man pages are included in this section:

- `backup(8)`—replacement
- `ctime(3)`—replacement
- `date(1)`—replacement
- `gettimeofday(3)`—new libc functionality
- `test(1)`—replacement
- `tzfile(5)`—new file format
- `tzset(3)`—new man page
- `tzsetup(8)`—new man page
- `zdump(8)`—new utility
- `zic(8)`—new utility

NAME

backup, bldboot, restore – backup/restore SPU disk files program

SYNOPSIS

/etc/backup [**-r**]
/etc/bldboot file1 file2 [**-r**] [**-f**]
/etc/restore

DESCRIPTION

Backup is a script which executes the commands necessary to backup the SPU disk. If the *-r* option is specified, only the root partition is backed up.

The tape drive used by **backup** and **restore** will be */dev/rmt1* (the QIC tape drive) if it exists. Otherwise, the Cipher tape drive will be used. Note that on sp2, the Cipher tape drive is not supported. For **bldboot**, the QIC tape drive will be used by default if */dev/rmt1* exists. However, the **-f** switch can be used to force the use of the Cipher tape drive (C-1 only). **Note that if a backup is to occur on a Cipher tape, the tape must be properly formatted prior to performing the backup.** Failure to do so will result in rather misleading error messages such as *error reading bad block table = -1.*

Bldboot is the program that is executed to backup the boot tracks and the root image in binary form and, if the *-r* is not specified, the mounted file system(s) in **tar**(1) format. *File1* is the copy of the cartridge tape boot track that is stored on the root file system. *File2* is the copy of the disk/tape format utility used to format and build the disk and tape on the SPU. The **bldboot** program will place the following data on the cartridge tape:

Four copies of the cartridge tape boot track program. These copies are used by the SPU EPROM program. The copies should be writable in about 25 seconds on the Cipher tape drive (C-1 only), or about 3 seconds on the QIC tape drive.

Two copies of the disk/tape format utility (spu2000). The tape boot track program will load copy 0 first and if errors are detected the second copy will be used, if possible. The copies should be writable in about a minute on the Cipher tape drive (C-1 only), or about 5 seconds on the QIC tape drive.

Two copies of the boot/root binary image. The boot tracks and the root file system are copied to the tape twice. The disk/tape format routine will load the copy the user selected. Each copy should be writable in about six minutes on the Cipher tape drive (C-1 only), or about one minute on the QIC tape drive.

The *tar* of the mount file system(s) will take an amount of time directly related to the number of files.

Restore will finish the rebuild procedure. **Restore** is a script that will build the mount file system(s) by executing a **mkfs**(8) and a **mklost+found**(8) on the mount partition(s). The **restore** script will then execute a **tar**(1) command to restore the mount file system(s). The time it takes to restore the file system will be related to how many files are on the tape.

FILES

/bt0.cart	the cartridge tape boot program
/stand/spu2000.cart	the disk/tape format utility
/dev/dk0a	boot tracks and root file system source (Winchester disk)
/dev/dk0d	mount system partition (Winchester disk)
/dev/dk0e	mount system partition 2 (sp2 only)
/dev/rct0b	the partition on the cartridge tape for boot/root backup (Cipher tape,

C-1 only)

/dev/rmt1	QIC tape drive
/mnt	the /dev/dk0d file system root node name
/hw	the /dev/dk0e file system root node name (sp2 only)
/dev/dk2a	boot tracks and root file system source (IOmega disk)
/dev/dk3a	mount file system partition (IOmega disk)
/dev/dk2d	mount file system partition (IOmega disk)

SEE ALSO

tar(1)
format(8)
backup(5)

DIAGNOSTICS

Bldboot enables the write verify mode in the SPU OS cartridge tape driver. This mode will cause the tape to write a block, rewind to the front of the block and then read the block just written. The write verify mode attempts to ensure that the tape written can be read with out errors.

BUGS

Backups on different versions of SPU OS may not be compatible. The boot tracks and the root partition are backed up in binary format.

WARNINGS

It is important to note that **backup** and **restore** are not general purpose utilities. They have specific knowledge of the SPU environment, and only backup and restore file systems that are expected to be there. Specifically, on C-1 systems, only the *root* and */mnt* partitions are backed up. (Older C-1 systems with Iomega drives have an additional partition, */mnt/os*, that is also backed up.) On C-2, C3200, and C3400 systems, only the *root*, */mnt*, and */hw* file systems are backed up. */tmp* is never backed up or restored. On C3400 systems, the */sst* and */scratch* file systems are never backed up or restored, either. On any system, additional file systems that have been created in unallocated space on larger-than-normal disks are never backed up. It is up to the user to perform his own backups on partitions that are not backed up.

Backup and **restore** do not use */etc/fstab* in their efforts. They expect particular mount points to be located on particular partitions. Changes to */etc/fstab* may cause undesirable behavior.

NAME

ctime, localtime, gmtime, asctime, timezone, mktime, strftime – date and time manipulation routines

SYNOPSIS

```
#include <time.h>

char *ctime(const time_t *clock);

struct tm *localtime(const time_t *clock);

struct tm *gmtime(const time_t *clock);

char *asctime(const struct tm *tm);

char *timezone(int zone, int dst);

time_t mktime(struct tm *timeptr);

size_t strftime(char *s, size_t maxsize, const char *format, const struct tm *timeptr);
```

DESCRIPTION

ctime converts a time pointed to by *clock* such as returned by *time*(3C) into ASCII and returns a pointer to a 26-character string in the following form. (All the fields have constant width.)

Sun Sep 16 01:03:52 1973\n\0

localtime and *gmtime* return pointers to structures containing the broken-down time. *localtime* corrects for the time zone and possible daylight savings time; *gmtime* converts directly to GMT, which is the time UNIX uses. (UNIX is a registered trademark of UNIX System Laboratories, Inc.) *asctime* converts a broken-down time to ASCII and returns a pointer to a 26-character string.

The structure declaration from the include file is:

```
struct tm {
        int tm_sec;     /* 0-59   seconds */
        int tm_min;     /* 0-59   minutes */
        int tm_hour;    /* 0-23   hour */
        int tm_mday;    /* 1-31   day of month */
        int tm_mon;     /* 0-11   month */
        int tm_year;    /* 0-     year - 1900 */
        int tm_wday;    /* 0-6    day of week (Sunday = 0) */
        int tm_yday;    /* 0-365  day of year */
        int tm_isdst;   /* flag:  daylight savings time in effect */
};
```

These quantities give the time on a 24-hour clock, day of month (1-31), month of year (0-11), day of week (Sunday = 0), year – 1900, day of year (0-365), and a flag that is nonzero if daylight saving time is in effect.

When local time is called for, the program consults the system to determine the time zone and whether the U.S.A., Australian, Eastern European, Middle European, or Western European daylight saving time adjustment is appropriate. The program knows about various peculiarities in time conversion over the past 10-20 years; if necessary, this understanding can be extended.

timezone returns the name of the time zone associated with its first argument, which is measured in minutes westward from Greenwich. If the second argument is 0, the standard name is used, otherwise the Daylight Saving version. If the required name does not appear in a table built into the routine, the difference from GMT is produced; e.g., in Afghanistan *timezone(-(60*4+30), 0)* is appropriate because it is 4:30 ahead of GMT and the string **GMT+4:30** is produced.

mktime converts the broken down local time, contained in a *struct tm* into a calendar time value with the same encoding as that of the values returned by the *time* function. The original values of the *tm_wday* and *tm_yday* components of the structure are ignored, and the original values of the other components are not restricted to the ranges indicated. This allows positive values of *tm_isdst* to indicate that Daylight Saving Time is initially in effect, zero to indicate that Daylight Saving Time is not initially in effect, and negative values indicate that *mktime* is to attempt to determine if Daylight Saving Time is in effect for the specified time.

Upon successful completion, the values of the *tm_wday* and *tm_yday* components of the structure are set appropriately, and the other components are set to represent the specified calendar time, but with their values forced to the ranges indicated above; the final value of *tm_mday* is not set until *tm_mon* and *tm_year* are determined.

strftime places characters in the array pointed to by *s* as controlled by the string pointed to by *format*. The format shall be a multibyte character sequence, beginning and ending in its initial shift state. The *format* string consists of zero or more conversion specifiers and ordinary multibyte characters. A conversion specifier consists of a % character followed by a character that determines the behavior of the conversion specifier. All ordinary multibyte characters (including the terminating null character) are copied unchanged into the array. If copying takes place between objects that overlap, the behavior is undefined. No more than *maxsize* characters are placed in the array. Each conversion specifier is replaced by appropriate characters as described in the following table. The appropriate characters are determined by the *LC_TIME* category of the current locale and by the values contained in the structure pointed to by *timeptr*.

Specifier	Definition
%a	locale's abbreviated weekday name.
%A	locale's full weekday name.
%b	locale's abbreviated month name.
%B	locale's full month name.
%c	locale's appropriate date and time representation.
%d	day of the month as a decimal number (01-31).
%H	hour (24 hour clock) as a decimal number (00-23).
%I	hour (12 hour clock) as a decimal number (01-12).
%j	day of the year as a decimal number (001-366).
%m	month as a decimal number (01-12).
%M	minute as a decimal number (00-59).
%p	locale's equivalent of the AM/PM designations associated with a 12 hour clock.
%S	second as a decimal number (00-61).
%U	week of the year as a decimal number (00-53, first Sunday as first day of week 1).
%w	weekday as a decimal number (0-6) where Sunday is 0.
%W	week of the year as a decimal number (00-53, first Monday as first day of week 1).
%x	locale's appropriate date representation.
%X	locale's appropriate time representation.
%y	year without century as a decimal number (00-99).
%Y	year with century as a decimal number.
%Z	time zone name or abbreviation, or empty string if there is no time zone information.
%%	the % character

If a conversion specifier is not one of the above, the behavior is undefined.

strftime returns the number of characters actually placed in the array, not including the null character, if the total number of characters, including the null character, is less than *maxsize*. Otherwise, zero is returned, and the contents of the array are indeterminate.

COMPATIBILITY

The *strftime* function is not available in the backward-compatible (-pcc) mode of CONVEX C.

SEE ALSO

gettimeofday(3), time(2), tzset(3)

BUGS

The return values of mktime, ctime, localtime, and gmtime, point to static data whose content is overwritten by each call.

NAME

date – print and set the date

SYNOPSIS

date [-u] [+*format*] [yymmddhhmm [.ss]]

DESCRIPTION

If no arguments are given, the current date and time are printed. If an operand begins with a "+" the output format of **date** shall be controlled by the format descriptors and text within the string *format*. If a date is specified, the current date is set. The *-u* flag is used to display the date in UTC (universal) time. This flag may also be used to set UTC time. *yy* is the last two digits of the year; the first *mm* is the month number; *dd* is the day number in the month; *hh* is the hour number (24 hour system); the second *mm* is the minute number; *.ss* is optional and is the seconds. For example:

date 10080045

sets the date to Oct 8, 12:45 AM. The year, month and day may be omitted, the current values being the defaults. The system operates in UTC. *Date* takes care of the conversion to and from local standard and daylight time.

The time zone mnemonics displayed by **date** can be modified by specifying the desired mnemonics in the environmental variable TZNAME. For example,

TZNAME="STD,DST"; export TZNAME

will result in STD and DST being displayed as the standard time and daylight savings time mnemonics, respectively.

The allowed field descriptors used in the *format* string are as follows:

%a	the abbreviated weekday name.
%A	the full weekday name.
%b	the abbreviated month name.
%B	the full month name.
%c	complete date and time representation.
%C	the current century (the current year, divided by 100, and truncated to two characters) as a decimal number (00-99).
%d	the day of the month, as a decimal number (01-31).
%D	the date in the format *mm/dd/yy*.
%e	the day of the month as a decimal number, space filled to two digits (1-31)
%h	a synonym for **%b**.
%H	the hour (24-hour clock) as a decimal number (00-23).
%I	the hour (12 hour clock) as a decimal number (01-12).
%j	the day of the year as a decimal number (001-366).
%m	the month as a decimal number (01-12).
%M	the minute of the hour as a decimal number (00-59).
%n	the newline character.
%p	**AM** or **PM** indicator.
%r	12-hour clock time (01-12) using *AM/PM* notation.

%S	seconds as a decimal number (00-61).
%t	a tab character
%T	24-hour clock time (00-23) in the format *HH:MM:SS*.
%U	week number of the year (Sunday as first day of the week) as a decimal number (00-53).
%w	weekday as a decimal number [0(Sunday)-6].
%W	week number of the year (Monday as the first day of the week) as a decimal number (00-53).
%x	locale's appropriate date representation.
%X	locale's appropriate time representation.
%y	the year (offset from **%C**) as a decimal number (00-99).
%Y	the year with century as a decimal number.
%Z	time-zone name, or the empty string if the time-zone is not determinable.
%%	a percent sign.

The **-z** option has been obsoleted. Refer to *zic*(8) to change time-zones.

FILES

/usr/adm/wtmp to record time-setting

SEE ALSO

utmp(5), ctime(3), tzset(3), zic(8)

CHANGES

This version of **date** can no longer set the systems idea of the time zone. The system time zone is now set using the **zic** utility.

DIAGNOSTICS

'Failed to set date: Not owner' if you try to change the date but are not the superuser.

NAME

gettimeofday, settimeofday – get/set date and time

SYNOPSIS

```
#include <sys/time.h>

gettimeofday(tp, tzp)
struct timeval *tp;
struct timezone *tzp;

settimeofday(tp, tzp)
struct timeval *tp;
struct timezone *tzp;
```

DESCRIPTION

Gettimeofday returns the system's notion of the current Greenwich time and the current time zone. Time returned is expressed relative in seconds and microseconds since midnight January 1, 1970.

The structures pointed to by *tp* and *tzp* are defined in *<usr/include/sys/time.h>* as:

```
struct timeval {
        long    tv_sec;         /* seconds since Jan. 1, 1970 */
        long    tv_usec;        /* and microseconds */
};

struct timezone {
        int     tz_minuteswest; /* of Greenwich */
        int     tz_dsttime;     /* type of dst correction to apply */
};
```

The *timezone* structure indicates the local time zone (measured in minutes of time westward from Greenwich), and a flag that, if nonzero, indicates that Daylight Saving time applies locally during the appropriate part of the year.

Only the super-user may set the time of day.

RETURN

A 0 return value indicates that the call succeeded. A –1 return value indicates an error occurred, and in this case an error code is stored into the global variable *errno*.

ERRORS

The following error codes may be set in *errno*:

[EFAULT] An argument address referenced invalid memory.

[EPERM] A user other than the super-user attempted to set the time.

NOTES

This is a libc emulation of the gettimeofday(2) system call found on BSD systems.

The *timeval* structure is only accurate to ten (10) microseconds, since that is the accuracy of the underlying software timer mechanism.

SEE ALSO

date(1), ctime(3)

NAME

test - condition command

SYNOPSIS

test expr

DESCRIPTION

Test evaluates the expression *expr*, and if its value is true then returns zero exit status; otherwise, a non zero exit status is returned. **Test** returns a non zero exit if there are no arguments.

The following primitives are used to construct *expr*.

–r file — true if the file exists and is readable.

–w file — true if the file exists and is writable.

–f file — true if the file exists and is not a directory.

–d file — true if the file exists and is a directory.

–s file — true if the file exists and has a size greater than zero.

–t [fildes]
: true if the open file whose file descriptor number is *fildes* (1 by default) is associated with a terminal device.

–z s1 — true if the length of string *s1* is zero.

–n s1 — true if the length of the string *s1* is nonzero.

s1 = s2 — true if the strings *s1* and *s2* are equal.

s1 != s2 — true if the strings *s1* and *s2* are not equal.

s1 — true if *s1* is not the null string.

n1 **–eq** n2
: true if the integers *n1* and *n2* are algebraically equal. Any of the comparisons **–ne**, **–gt**, **–ge**, **–lt**, or **–le** may be used in place of **–eq**.

These primaries may be combined with the following operators:

! — unary negation operator

–a — binary *and* operator

–o — binary *or* operator

(expr)
: parentheses for grouping.

–a has higher precedence than **–o.** Notice that all the operators and flags are separate arguments to **test.** Notice also that parentheses are meaningful to the Shell and must be escaped.

SEE ALSO

sh(1)

NAME

tzfile – time zone information

SYNOPSIS

#include <tzfile.h>

DESCRIPTION

The time zone information files used by *tzset*(3) begin with bytes reserved for future use, followed by three four-byte values of type **long**, written in a "standard" byte order (the high-order byte of the value is written first). These values are, in order:

tzh_timecnt

The number of "transition times" for which data is stored in the file.

tzh_typecnt

The number of "local time types" for which data is stored in the file (must not be zero).

tzh_charcnt

The number of characters of "time zone abbreviation strings" stored in the file.

The above header is followed by *tzh_timecnt* four-byte values of type **long**, sorted in ascending order. These values are written in "standard" byte order. Each is used as a transition time (as returned by *time*(2)) at which the rules for computing local time change. Next come *tzh_timecnt* one-byte values of type **unsigned char**; each one tells which of the different types of "local time" types described in the file is associated with the same-indexed transition time. These values serve as indices into an array of *ttinfo* structures that appears next in the file; these structures are defined as follows:

```
struct ttinfo {
        long            tt_gmtoff;
        int             tt_isdst;
        unsigned int    tt_abbrind;
};
```

Each structure is written as a four-byte value for *tt_gmtoff* of type **long**, in a standard byte order, followed by a one-byte value for *tt_isdst* and a one-byte value for *tt_abbrind*. In each structure, *tt_gmtoff* gives the number of seconds to be added to GMT, *tt_isdst* tells whether *tm_isdst* should be set by *localtime (3)* and *tt_abbrind* serves as an index into the array of time zone abbreviation characters that follow the *ttinfo* structure(s) in the file.

Localtime uses the first standard-time *ttinfo* structure in the file (or simply the first *ttinfo* structure in the absence of a standard-time structure) if either *tzh_timecnt* is zero or the time argument is less than the first transition time recorded in the file.

SEE ALSO

ctime(3)

NAME

tzset – set time zone information

SYNOPSIS

#include <time.h>

void tzset()

DESCRIPTION

The *tzset()* function uses the value of the environment variable **TZ** to set time conversion information used by *localtime(), ctime(), strftime(),* and *mktime()*. If **TZ** is absent from the environment, implementation-defined default time zone information is used.

The *tzset()* function sets the external variable *tzname:*

extern char *tzname[2] = { "*std*", "*dst*" };

where *std* and *dst* are the standard and summer time zone names defined form the environment variable **TZ**.

The value of **TZ** has the following form (spaces have been inserted for clarity):

std offset dst offset, rule

Where:

std and *dst* are three or more byte sequences that are the designation for the standard (*std*) or summer (*dst*) time zone. Only *std* is required; if *dst* is missing, then summer time does not apply in this locale. Upper- and lowercase letters are explicitly allowed. Any characters except a leading colon (:), digits, comma(,), minus(-), plus(+), and ASCII NULL are allowed.

offset indicates the value one must add to the local time to arrive at Coordinated Universal Time. The *offset* has the form:

hh[:mm[:ss]]

The minutes (*mm*) and seconds (*ss*) are optional. The hour (*hh*) is required and may be a single digit. The *offset* following *std* is required. If no *offset* follows *dst,* summer time is assumed to be one hour ahead of standard time. One or more digits may be used; the value is always interpreted as a decimal number. The hour shall be between zero and 24, and the minutes (and seconds) - if present - between zero and 59. Out of range values will cause unpredictable behavior. If preceded by a '-', the time zone is east of the Prime Meridian; otherwise it is west (which may be indicated by an optional preceding '+').

Rule indicated when to change to and back from summer time. The *rule* has the form:

date/time,date/time

where the first *date* describes when the change from standard to summer occurs, and the second *date* describes when the change back happens. Each *time* field describes when, in current local time, the change to the other time is made. The format of *date* shall be one of the following:

J*n*, the Julian day *n* (one based). Because leap days are not counted, it is impossible to refer to February 29.

n, The zero-based Julian day. Because leap days are counted, it is possible to refer to February 29.

M*m.n.d*, day *d* of week *n* of month *m*. Week *n* is the *n*th week in which day *d* appears. Week 5 is always the last week. Sunday is day zero.

The *time* has the same format as *offset* except that no leading sign ('-' or '+') is allowed. The default, if *time* is not given, is 02:00:00.

NAME

tzsetup – set up old-style time zone information in the SPU OS kernel

SYNOPSIS

tzsetup

DESCRIPTION

Tzsetup attempts to find the offset from GMT for the current timezone based on the default timezone pointed to by */etc/zoneinfo/localtime*. It does this by calculating the difference between GMT and a time of the year when daylight savings time (DST) is not in effect. It also determines if DST is ever in effect during the year. Both of these bits of information are then updated in the SPU OS kernel so that utilities that were not linked with the new, *zic* compatible libraries, can report the correct date and time on the SPU. The update occurs using a *settimeofday(8)* function.

FILES

/etc/zoneinfo/localtime hard link to the default timezone.

SEE ALSO

gettimeofday(3), tzfile(5), zdump(8), zic(8)

NAME

zdump – time zone dumper

SYNOPSIS

zdump [**–v**] [**–c** cutoffyear] [zonename ...]

DESCRIPTION

Zdump prints the current time in each *zonename* named on the command line.

These options are available:

–v For each *zonename* on the command line, print the current time, the time at the lowest possible time value, the time one day after the lowest possible time value, the times both one second before and exactly at each time at which the rules for computing local time change, the time at the highest possible time value, and the time at one day less than the highest possible time value. Each line ends with **isdst=1** if the given time is Daylight Saving Time or **isdst=0** otherwise.

–c *cutoffyear*
Cut off the verbose output near the start of the given year.

FILES

/etc/zoneinfo standard zone information directory

SEE ALSO

ctime(3), tzfile(5), zic(8)

NAME

zic – time zone compiler

SYNOPSIS

zic [**–v**] [**–d** *directory*] [**–l** *localtime*] [*filename ...*]

DESCRIPTION

Zic reads text from the file(s) named on the command line and creates the time conversion information files specified in this input. If a *filename* is –, the standard input is read.

These options are available:

–d *directory*

Create time conversion information files in the named directory rather than in the standard directory named below.

–l *timezone*

Use the given time zone as local time. *Zic* will act as if the file contained a link line of the form

Link *timezone* localtime

–v Complain if a year that appears in a data file is outside the range of years representable by *time*(2) values.

Input lines are made up of fields. Fields are separated from one another by any number of white space characters. Leading and trailing white space on input lines is ignored. An unquoted sharp character (#) in the input introduces a comment which extends to the end of the line the sharp character appears on. White space characters and sharp characters may be enclosed in double quotes (") if they're to be used as part of a field. Any line that is blank (after comment stripping) is ignored. Non-blank lines are expected to be of one of three types: rule lines, zone lines, and link lines.

A rule line has the form

Rule	NAME	FROM	TO	TYPE	IN	ON	AT	SAVE	LETTER/S

For example:

Rule	USA	1969	1973	–	Apr	lastSun	2:00	1:00	D

The fields that make up a rule line are:

NAME Gives the (arbitrary) name of the set of rules this rule is part of.

FROM Gives the first year in which the rule applies. The word **minimum** (or an abbreviation) means the minimum year with a representable time value. The word **maximum** (or an abbreviation) means the maximum year with a representable time value.

TO Gives the final year in which the rule applies. In addition to **minimum** and **maximum** (as above), the word **only** (or an abbreviation) may be used to repeat the value of the **FROM** field.

TYPE Gives the type of year in which the rule applies. If **TYPE** is – then the rule applies in all years between **FROM** and **TO** inclusive; if **TYPE** is **uspres**, the rule applies in U.S. Presidential election years; if **TYPE** is **nonpres**, the rule applies in years other than U.S. Presidential election years. If **TYPE** is something else, then *zic* executes the command

yearistype *year type*

to check the type of a year: an exit status of zero is taken to mean that the

year is of the given type; an exit status of one is taken to mean that the year is not of the given type.

IN Names the month in which the rule takes effect. Month names may be abbreviated.

ON Gives the day on which the rule takes effect. Recognized forms include:

5	the fifth of the month
lastSun	the last Sunday in the month
lastMon	the last Monday in the month
Sun>=8	first Sunday on or after the eighth
Sun<=25	last Sunday on or before the 25th

Names of days of the week may be abbreviated or spelled out in full. Note that there must be no spaces within the **ON** field.

AT Gives the time of day at which the rule takes effect. Recognized forms include:

2	time in hours
2:00	time in hours and minutes
15:00	24-hour format time (for times after noon)
1:28:14	time in hours, minutes, and seconds

Any of these forms may be followed by the letter **w** if the given time is local "wall clock" time or **s** if the given time is local "standard" time; in the absence of **w** or **s**, wall clock time is assumed.

SAVE Gives the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the **AT** field (although, of course, the **w** and **s** suffixes are not used).

LETTER/S
Gives the "variable part" (for example, the "S" or "D" in "EST" or "EDT") of time zone abbreviations to be used when this rule is in effect. If this field is –, the variable part is null.

A zone line has the form

```
Zone  NAME                GMTOFF  RULES/SAVE  FORMAT  [UNTIL]
```

For example:

```
Zone  Australia/South-west  9:30  Aus         CST     1987 Mar 15 2:00
```

The fields that make up a zone line are:

NAME The name of the time zone. This is the name used in creating the time conversion information file for the zone.

GMTOFF
The amount of time to add to GMT to get standard time in this zone. This field has the same format as the **AT** and **SAVE** fields of rule lines; begin the field with a minus sign if time must be subtracted from GMT.

RULES/SAVE
The name of the rule(s) that apply in the time zone or, alternately, an amount of time to add to local standard time. If this field is – then standard time always applies in the time zone.

FORMAT

The format for time zone abbreviations in this time zone. The pair of characters %s is used to show where the "variable part" of the time zone abbreviation goes.
UNTIL The time at which the GMT offset or the rule(s) change for a location. It is specified as a year, a month, a day, and a time of day. If this is specified, the time zone information is generated from the given GMT offset and rule change until the time specified.

The next line must be a "continuation" line; this has the same form as a zone line except that the string "Zone" and the name are omitted, as the continüation line will place information starting at the time specified as the **UNTIL** field in the previous line in the file used by the previous line. Continuation lines may contain an **UNTIL** field, just as zone lines do, indicating that the next line is a further continuation.

A link line has the form

Link LINK-FROM LINK-TO

For example:

Link US/Eastern EST5EDT

The **LINK-FROM** field should appear as the **NAME** field in some zone line; the **LINK-TO** field is used as an alternate name for that zone.

Except for continuation lines, lines may appear in any order in the input.

NOTE

For areas with more than two types of local time, you may need to use local standard time in the **AT** field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

FILES

/etc/zoneinfo standard directory used for created files

SEE ALSO

ctime(3), tzfile(5), zdump(8)