Unirex

System Administrator's Guide

Dansk Data Elektronik A/S
27 July 1983

Author: Claus Tøndering

Table of Contents.

1. Introduction

This manual describes how the Unirex operating sytem is maintained. The job of the System Administrator and the tools at his or her disposal are described.

The reader is assumed to be acquainted with the Unirex Introductory Guide and the Unirex System Operation Guide.

2. The System Administrator.

The System Administrator is the person at a computer installation responsible for the maintenance of the computer. Of course, many of the tasks here assigned to the System Administrator may easily be performed by several persons at a particular installation, but for ease of description the following chapters will be based on the assumption that there is only one System Administrator.

The System Administrator's tasks include the following:

1) Ensuring an adquate bootstrapping of the computer.
2) Maintaining the list of persons that may use the computer.
3) Perform backups of the sytem disks.
4) Take care of system crashes.

3. Bootstrapping.

Bootstrapping is the process of bringing the computer up after a power down period or a system crash.

The bootstrapping process runs in the following manner:

1) A reset signal is given to the CPUs. This happens in the following two cases:

   a) When power is applied to the computer.
   b) When the key on the chassis containing the CPUs is turned.

2) The display on the chassis will show 'b0' for all MCUs, indicating that they are bootstrapping.

3) The computer loads files from a floppy disk. The files loaded are

scmikm-0	which is loaded into the SIOCs, that is, the CPUs that control terminals and printers.
dcmikm-0	which is loaded into the DIOCs, that is, the CPUs that control disks.
mcub-6	which is loaded into the MCUs.

4) When mcub-6 has been loaded into an MCU, this MCU starts executing the initialization part of Unirex, during which period 'c0' is displayed on the CPU chassis.

5) The display is turned off. Now Unirex is up and running. The access rights of all terminals and printers are set to: ---rw-rw-rw-. The access rights of all disks are set to: ---rw-r--r--. The owner of all terminals, printers, and disks is set to zero.

6) The MCUs execute the file /initN-1, where the letter N is replaced by the MCU number; thus MCU number 0 executes /init0-1, MCU number 3 executes /init3-1 etc. These programs execute with the user number set to zero. Each of these programs execute vox with input from the file /initN-k, where the letter N is replaced by the MCU number. Thus /init0-k contains a command file that is executed in MCU number 0, the commands in /init3-k are executed in MCU number 3 etc. If any error occurs during the execution of these command files, the error message will be displayed on :term0.

During bootstrapping an error may occur, perhaps accompanied by a system crash (see chapter 6). In this case an error value will be displayed on the MCU display on the computer. The error values are found in appendix A.

3.1. The init command files.

The command files /init0-k, /init1-k, etc. should contain the following if the logon mechanism is to work properly.

```
install file /logon-1
create unit :combox/mcuN size 80 prot 0660
@vox input :combox/mcuN list :combox/mcuN error :null output :null
```

In the above lines N should be replaced by the number of the MCU in which the command file is executed.

In one of the MCUs the logon process for each terminal should be started. As an MCU with a high number is bootstrapped before an MCU with a low number, it is generally good practice to start the logon processes in the MCU with the lowest number, generally MCU number zero.

The file /init0-k should therefore include the following lines after the lines given above:

```
@logon :term0 error :null input :null
@logon :term1 error :null input :null
```

And so on for each terminal from which logon should be possible.

The init files may also typically contain commands that perform the following:

- set the baud rate and other characteristics of various terminals and printers.
- install various often-used programs (vox or inter, for example).
- mount various often-used disks.
- request the System Adminstrator to set the system clock.

- change the access rights and ownership of certain dedicated disks, terminals, or printers.

4. User and Logon Administration.

This chapter describes how the System Adminstrator sets up the logon picture and admits users to the computer.

The key to the user and logon administration is a text file called /users-k. This file contains a specification of the text that is to be displayed on the terminal during logon, plus a list of all the users that may access the computer.

A typical /users-k file is shown on the following page.

```
<'    'X>
<C0902>**************************************************************************
<C0903'*'dl'*'dl'*'dl'*'dl'*'dl'*'dl'*'dl'*'>
<C7310'*'ul'*'ul'*'ul'*'ul'*'ul'*'ul'*'ul'*'>
<C0911>**************************************************************************
<C3304>Welcome to the
<C1206>multi-tasking, multi-user, multi-processor
<C5506>operating system
.
<C3008>UNIREX version
<C5310>Terminal number:
<C1216>Enter user name:
<C4316Z>Enter password:
<C2118>You are logging on to processor number:
<C2120Z'Ok.'C0124>
<C2120>This processor is not present.
<C2121>ERROR IN STARTING PROCESS!
<C5916>Accepted
<C5916>Rejected
.
 1              16              32              48              64      72
 NAME           PASSWORD        INS PROG.       MCU. MASK       USER£   LOAD

*dde            priv            vox             0xffff          0x0000  4
/vox-1
:disk0/

*mickey         mouse           comal           0xffff          0x0101  4
/comal-1
:disk0/

*donald         duck            inter           0xffff          0x03a5  4
/inter-1 /edit,/thisfile
:disk1/

-
```

This file consists of three parts. The first part starts at the first line and runs through the line containing a single period. The second part starts after this period and runs through the second line containing a single period. The third part starts after this period and runs through the final hyphen.

The first part contains lines that are to be displayed as the logon picture. These lines must start with a <, or else they will be considered comments. There may be any number of lines in this part of the file.

The second part must contain ten lines:

- The first line ('<C3008>UNIREX version' in the above example) will be displayed as part of the logon picture, followed by the Unirex version date.
- The second line ('<C5310>Terminal number:' in the above example) will be displayed as part of the logon picture, followed by the terminal number.
- The third line ('<C1216>Enter user name:' in the above example) will be displayed when the logon program prompts the user to enter the user name.
- The fourth line ('<C4316Z>Enter password:' in the above example) will be displayed when the logon program prompts the user to enter the password.
- The fifth line ('<C2118>You are logging on to processor number:' in the above example) will be displayed when the logon program prompts the user to enter the number of the MCU on which the user's programs will run.
- The sixth line ('<C2120Z'Ok.'C0124>' in the above example) will be displayed when an attempt to logon has been successfull.
- The seventh line ('<C2120>This processor is not present.' in the above example) will be displayed if the user attempts to log on to an MCU that is not installed in the computer.
- The eighth line ('<C2121>ERROR IN STARTING PROCESS!' in the above example) will be displayed, if for some reason the logon program does not receive information from the MCU to which the user logs on, that the user program is started. A probable cause may be that there is no memory available in the target MCU.
- The ninth line ('<C5916>Accepted' in the above example) will be displayed when a valid user name an password have been entered.
- The tenth line ('<C5916>Rejected' in the above example) will be displayed when an invalid user name or password has been entered.

The third part of the file contains a list of all the users that have access to the computer. Here, lines starting with blanks are mere comments. The lines

```
1               16              32              48              64      72
NAME            PASSWORD        INS PROG.       MCU. MASK       USER£   LOAD
```

in the above example therefore merely serve as a header for the following entries. Each user entry consists of three lines.

If we let the first position of a line be position zero, the format of the first line is:

Position 0: An asterisk.

Position 1: The user name (at most 8 characters).

Position 16: The password (at most 8 characters).

Position 32: The name of an installed program (at most 8 characters). The logon program will try to execute this installed program, if present, as the user's operator service program. Parameters for this program are found in the next line, as described below.

Position 48: A hexadecimal number called the MCU mask. The number is preceded by the characters '0x' which is the C language indication that this is a hexadecimal number. The MCU mask specifies which MCUs the user may use. This feature is not yet implemented

Position 64: A hexadecimal number being the user number, preceded byt the characters '0x'.

Position 72: A decimal number called the load. This number is a relative measure of how heavy a load the user is expected to exercise on the MCU. The logon program will try to spread out the load on all the MCUs in the computer. This feature is not yet implemented, all users are assumed to have the same load.

The second line contains, starting in position 0, the name of a program file, followed by a space, followed by an optional parameter

string. The logon program will try to execute this program if an attempt to start the installed program specified in position 32 of the previous line fails. The parameter string given in this second line is also used in the attempt to start the installed program.

The third line contains the user's initial current unit prefix.

In the above sample, the user 'donald' has the password 'duck'. His user number is 03a5, and when he logs on, the logon program will try to execute the installed program 'inter' giving it the parameter string '/edit,/thisfile'. If there is no installed program called 'inter', the logon program will try to load and execute the program in the file '/inter-l', giving it the parameter string '/edit,/thisfile'. The current unit prefix will be set to ':disk1/'.

4.1. How Does Logon Work?

When the computer is bootstrapped, a logon program is started for each terminal from which logon should be possible. When a user attempts to log on, the following happens (slightly simplified):

1) The logon program finds the MCU with the lowest current load (the fewest users logged on), and suggests that the user log on to this MCU. The load on each MCU is stored in a file called /ldfile-d. Let us assume that the logon program suggests MCU number 5, and that the user accepts this.

2) The logon program sends a command line to :combox/mcu5. This common box has been created by the initial command file that ran on MCU number 5 just after the bootstrapping.

3) A vox is present in MCU 5 (started also by the initial command file on this MCU), taking commands from :combox/mcu5. This vox will receive the command from the logon program. This command is a request to start a logon program on MCU 5.

4) The logon program on MCU 5 now spawns the operator service program on MCU 5. If successfull, a reply is sent to the original logon program through another common box.

5) The original logon program dies.

6) The logon program on MCU 5 waits for all its offspring processes to die. Normally, the logon program has only one offspring process, namely the operator service program; but it is possible that this program has gemmated other processes, in which case the logon program will have more than one offspring process. When all the offspring processes are dead, the user's session at the terminal is over.

7) The logon program on MCU 5 presents the logon picture and awaits a logon request from a new user.

5. Disk Backup.

It is good practice frequently to take a copy of the contents of all disks to some removable medium. Although this may seem hardly worth the trouble most of the time, it may some day save many days' work. The importance of backing up disks can hardly be overemphasized.

There are three ways to back up disk contents:

1) Using the diskcopy program to copy the contents of a disk onto a streamer tape.

2) Using the wback program to copy the contents of a disk onto a set of floppy disks.

3) File-by-file copy of files that have been changed since the last backup was made.

If an error occurs, there are two possibilities: Either it is desired to restore a complete disk, or it is desired to restore a single file.

The following should be done if the complete disk should be restored:

1) If backup was made to a streamer tape using the diskcopy program, the diskcopy program may be used to restore the contents of the disk.

2) If backup was made to a set of floppy disks using the wback program, the wback program may be used to restore the contents of the disk.

3) If backup was made file-by-file, a file-by-file restore must be made.

The following should be done if a single file should be restored:

1) If backup was made to a streamer tape using the diskcopy program, the copy program may be used to copy the file from the streamer tape to the disk. Note that copying a single file from a streamer tape may take a very long time. Note also, that under no circumstances should anything but a complete disk backup be written to a streamer tape. Anything written to the streamer tape will be writ-

ten at the start of the streamer tape, regardless of the desired write position. This also implies, that access rights to files on streamer tape cannot be changed.

2) If backup was made to a set of floppy disks using the wback program, the procedure for restoring a single file is complicated. The floppy disks will have no directly understandable file structure, therefore it is not possible to copy a single file. Instead, the following procedure may be used. Let us assume that the file 'alpha-k' should be restored on :disk1. Let us call the set of backup disks 'disk set A'.

   - Take a complete backup of :disk1 to another set of floppy disks; we will call this set of backup disks 'disk set B'.
   - Restore the complete old contents of :disk1 from disk set A using the wback program.
   - Copy :disk1/alpha-k to :disk0/alpha-k (or some other disk).
   - Restore the complete contents of :disk1 from disk set B using the wback program.
   - Copy :disk0/alpha-k to :disk1/alpha-k.
   - Delete :disk0/alpha-k.

3) If file-by-file backup was made, the file may simply be restored.

6. System Crash.

Nobody is infallible. Error-free programs are as rare as oases in a desert, or perhaps even more so. Therefore it is inevitable that now and then an error in Unirex occurs. The system has been designed in such a way that it often is able to dectect internal errors and stop, rather than continue erroneous execution. This is known as a system crash.

When the system crashes the following happens:

1) An error code is displayed on the MCU display on the computer. These error codes are found in appendix A.

2) The following text appears on :term0:

```
**********************
**** SYSTEM CRASH ****
**********************

INSERT FLOPPY DISK ON WHICH TO DUMP MEMORY
NOTE: PREVIOUS DISK CONTENTS WILL BE DESTROYED!
PRESS RETURN:
```

3) A floppy disk should be inserted into the floppy disk drive and the return key on :term0 should be pressed. Any previous contents on the floppy disk will be destroyed.

4) The computer will now dump the contents of the Unirex data area on the disk along with information about what caused the system crash.

5) When memory has been dumped, the letters FF are displayed in the MCU display, the supervisor/user mode lamps are both switched off, and the text

```
DUMP TERMINATED - UNIMAX HALTED - REBOOT
```

will be shown on :term0.

6) Now the computer should be bootstrapped. Note that on other MCUs there may still be users working, and they may wish to save their work before the system is bootstrapped.

7) Send the floppy disk with the dump to us. From it we may be able to find out what happened and correct the error. This will be a great help both to us and to our customers.

Appendix A. Display Values.

The Unimax computer is equipped with a 2-digit hexadecimal display for each MCU. Normally this display is off, but during bootstrapping, and when an error occurs within Unirex, a value is displayed. The following table lists the possible values. Many of the explanations given may seem cryptic, but as this information is mainly relevant to our programmers, this should not trouble the reader. If the value below is preceded by an asterisk, a system crash accompanies the displaying of the value. Please note that the hexadecimal digits a-f are displayed in the following manner: A b C d E F. It is especially important to notice the difference between b and 6.

Value	Reason
* 00	Bus error within Unirex.
* 01	Address error (odd address) within Unirex.
* 02	Illegal instruction within Unirex.
* 03	Division by zero within Unirex.
* 04	CHK instruction failure within Unirex.
* 05	TRAPV instruction executed within Unirex.
* 07	Trace trap within Unirex.
* 08	Line 1010 emulator trap within Unirex.
* 09	Line 1111 emulator trap within Unirex.
* 0A	Illegal trap within Unirex.
12	Illegal common box interrupt.
13	Illegal interrupt.
* 14	Bad call of ipmovepcb in process management.
* 15	No active process.
* 16	Address in delitem or delcoitem call outside data area range.
* 18	Supervisor stack overflow.
* 19	Bad call of ipmovepcb in box management.
* 20	Attempt to release a not-reserved file.
b0	The MCU is bootstrapping.
C0	Unirex is initializing.
E5	Unirex bootstrap error: Cannot start init process.

*	E6	Unirex bootstrap error: Cannot start system process.
*	E7	Unirex bootstrap error: Cannot initialize partitions.
	E8	Unirex bootstrap error: Erroneous hardware configuration file.
	E9	Unirex bootstrap error: Specified and actual memory differ.
*	Ed	Unirex bootstrap error: Cannot open :term0.
*	F1	Unirex bootstrap error: Cannot start system process.
*	F3	Unirex bootstrap error: No common memory defined.
*	F4	Unirex bootstrap error: No file reservation memory defined.
*	F7	Odd address in creaitem or creacoitem call.
*	F8	Bad call of ipmovepcb in insipar routine.
	FF	Unimax halted after system crash dump.