Unirex

Utility Guide

<u>Preliminary</u>

Danish Data Electronic A/S

August 8, 1983

Author: Anders Ansted

Copyright 1983
Danish Data Electronic A/S

Contents

Chapter page

1. Introduction

This manual contains a preliminary description of a number of utility programs running under the Unirex operating system on the Unimax computer.

The manual describes the function and the parameters for each utility program, but does not describe system library routines.

Unirex and Unimax are registered trademarks of Danish Data Electronics A/S.

Danish Data Electronics A/S reserves the right to change the specifications in this manual without warning. Danish Data Electronics A/S is not responsible for the effects of typographical errors and other inaccuracies in this manual, and cannot be held liable for the effects of the implementations and use of the programs described herein.

2. Terminology

In this manual the number sign, , will be printed as £.

Optional parameters are enclosed in '()' brackets.

An iounit is an i/o unit, that is, any device or file with which input/output may be performed. Iounits include: The null device, terminals, printers, boxes of various kinds, disks, and files on disks.

An ioud is an iounit descriptor, that is, a short integer being the representation within the process of an open iounit.

3. General Information

3.1 Iounit names

An iounit name is a string of characters not including spaces. An iounit name uniquely identifies an iounit.

An iounit name comprises the name of the computer, where the unit resides, the device name and the name of the unit. The format for an iounit name is

```
        (((!computer):device)/)name/name/name
```

computer is the name of the computer on which the iounit resides. If omitted, 'this computer' is assumed.

device is the name of the device on which the iounit resides. This specification may be:

1) :null for the null device
2) :term0, :term1, etc. for terminal number 0, 1, etc.
3) :print0, :print1, etc. for printer number 0, 1, etc.
4) :disk0, :disk1, etc. for disk number 0, 1, etc. and for files residing on those disks
5) :box for boxes
6) :sysbox for system boxes
7) :combox for common boxes

If :device is omitted, ':disk0' is assumed.

/name/name/name is the specification of a file on a disk or the name of a box, a system box or a common box.

All characters in the iounit names are ideally lower case letters. Upper case letters are converted to the lower case counterparts.

If the first letter of an iounit name is not an exclamation mark, a colon or a slash, the iounit name is prefixed by the current unit prefix (set by vox). If, for example, the current unit prefix set by the vox command '.cup' is ':disk2/alpha/beta/' and the specified iounit name is 'gamma/delta' the effective iounit name will be ':disk2/alpha/beta/gamma/delta'.

3.2 Wildcards

Unless otherwise specified the last name component of a file name may comprise certain wildcard characters.

A wildcard character may be '*', '?' or a character interval enclosed in '<>' brackets. The format for a character interval is

        <first_character-last_character>.

A character interval includes the first and the last character, e.g. <d-f> means the characters 'd', 'e', and 'f', and the first_character must be less than the last_character.

The wildcard characters is used to specify 'don't cares', that is, an arbitrary character or string of characters. For example, the strings 'abc*' and 'abchhhjjjjii' are said to be equal, since the wildcard character '*' matches zero or more characters. The precise value of a wildcard character is:

        * matches zero or more characters
        ? matches a single character
        <first_character-last-character> matches any single character in the interval.

At most 10 wildcard characters are allowed in a name component.

If a program takes two iounitnames as parameters, which should in some way be matched or used in some sort of a name conversion, there must be a one-to-one wildcard character correspondance. An iounitname like ':disk1/a<b-d>s*' should be matched by an iounit name like e.g. ':disk1/q<b-d>x*'.

3.3 Standard iounit names.

As a convenient shorthand for 'my own standard input/output/error/list device' the special iounit names '£i', '£o', '£e' and '£l' may be used. Usually the standard input device is the terminal, and using the shorthand '£i' the terminal operator need not know the device name of the terminal.

3.3 Parameter/dialog mode

All utility programs described in this manual, except vox, may execute in either parameter mode or in dialog mode.

Running in parameter mode the parameters are given as program parameters (see the description of vox in this manual) at the time the program is invoked. That is, the parameters are passed to the program by the intervention of vox.

If a program requires more than one parameter, the individual parameters are qualified by a keyword. E.g. the program "rename a file" requires the old filename and the new filename. To give the file alpha of type k on disk0 the name beta, rename should be invoked like

```
        vox>rename from /alpha-k to /beta-k
```

The strings "from" and "to" are said to be keywords and are used to qualify the old and the new filename respectively, thus making the parameters position independent.

Qualified parameters may be given in any order, the only limitation is that the qualified parameter must follow the qualifying keyword immidiately.

If a program takes only one parameter, this parameters is not qualified, that is, there is no keyword.

Note that though many parameters may take default values and thus be omitted, the keyword for a possible single remaining parameter cannot be omitted.

Running a program in dialog mode, the utility program prompts for the necessary parameters. Generally, the programs show the best guess for a parameter value and the terminal operator just has to edit the proposed value and/or press the return key.

The utility programs send a message like "delete terminated" to the standard output unit when they terminate. If the termination is due to exceptional events, this fact is communicated as well.

3.4 Delimitors

Spaces are regarded as general delimitors. Spaces separate programname, program parameters, and standard iounit list elements.

Spaces are not significant characters in the sense that a string of spaces will be functionally equivalent to one space character.

4. Abort - Abort Process

This program is used to abort a running processes.

4.1 Description

Abort takes two parameters: processidentification and completion code. The abort program terminates the identified process with the given completion code.

A processidentification may be given either by the name of the process and the user number or by the processnumber.

Only a privileged user can abort a process belonging to another user.

4.1.1 Parameter mode

In parameter mode the format is

```
        abort processidentification (completion code)
```

and the keywords are:

```
        Processidentification:
             a) Processname:   name
                Usernumber:    user (default value: the users number)
             b) Processnumber: number
        Completion code:       cc (default value: 0)
```

4.1.1.1 Examples

```
vox>abort name comal
```

Aborts the users process named comal with completion code 0.

```
vox>abort number 823 cc 2
```

Aborts the process numbered 823 with completion code 2. Notice that the user must be the owner of that process or a privileged user, otherwise the abortion is refused.

4.1.2 Dialog mode

Running abort in dialog mode the dialog sequence is (user input is shown underlined):

```
Process (name or £procesnumber): comal
User number:                     4714
Completion code:                 0
```

or

```
Process (name or £procesnumber): £823
Completion code:                 2
```

5. Chaccess - Change Access Rights

Every iounit, except the :null device, has an access right description and an owner. The access right description consists of three parts: access rights for the owner, access rights for everyone else in the owners group, and access rights for everyone else in the system. It is posible to specify a combination of read, write and execute access to each of the three parts.

Besides, the access rights description contains the t-bit (postmortem dump on), the g-bit (set groupid on execution), and the u-bit (set userid on execution).

When a user wants to access an iounit, it will be checked if the user is identical to the owner, in the owners group, or outside the group, in order to select the right part of the access rights.

The chaccess program is used to change the access rights of a given iounit, and the t-, g-, and u-bit. Only a privileged user or the owner of the iounit are allowed to change the access rights

5.1 Description

The actual access rights for the specified iounit is found and updated. If so desired the updated access rights are restored.

The access rights for the owner, the owners group and others may be updated seperately. For each part any combination of r - read, w - write and x - execute may be specified.

5.1.1 Parameter Mode

In parameter mode the format is

```
      chaccess iounit (owner) (group) (others) (t-bit) (g-bit) (u-bit)
```

where t-bit, g-bit, and u-bit are either y (yes) or n (no) for the bit set or not set, respectively.

The keywords are

```
        iounit: unit
        owner: owner (default value is actual owner rights)
        group: group (default value is actual group rights)
        others: others (default value is actual others rights)
        dump: t-bit on (default n)
        sgroup: g-bit on (default n)
        suser: u-bit on (default n)
```

Note, use '-' if an empty access right should be specified.

5.1.1.1 Example

```
vox>>chaccess unit :print0 group rw others -
```

changes the group access rights of the :print0 device to read and write and deletes the access for other users.

5.1.2 Dialog Mode

In dialog mode the dialog is (user input underlined)

```
        Unit name: :print0
        The accesscodes are w - write, r - read, x - execute.
        Access rights for owner:         wr
        Access rights for group:         wr
        Access rights for others:        ___
        Postmortem dump on, t-bit (y/n): n
        Set groupid bit on, g-bit (y/n): n
        Set userid bit on, u-bit (y/n):  n
        Is the access still to be changed (y/n): y
```

6 Chown - Change Owner

The chown program is used to change the owner of a given iounit.

6.1 Description

The owner of the iounit is changed. The number of new owner is given as a hexadecimal number.

Only a privileged user is allowed to change the owner of an iounit.

6.1.1 Parameter Mode

In parameter mode the format is

```
        chown iounit owner
```

and the keywords are

```
        iounit: unit
        owner:  owner
```

6.1.1.1 Example

```
vox>>chown unit :print0 owner 50a.
```

change the owner of :print0 to be user a from group 5.

6.1.2 Dialog Mode

In dialog mode the dialog is (user input underlined)

```
        Unit name: :print0
        New owner: 50a
```

7 Compr - Disk Compression

This program is used to compress a disk, that is, to regain diskspace from deleted files. The program should be applied to Mikfile disks only.

7.1 Description

Compression of a disk has no influence on file contents. Unused filespace is not recovered by compr (use copy for this purpose).

Compr takes one parameter: the iounit name of the disk.

The disk must be mounted prior to compression.

7.1.1 Parameter mode

In parameter mode the format is

```
        compr iounit
```

No keyword.

7.1.1.1 Example

```
vox>compr :disk1
```

Compress the disk with the unit name :disk1.

7.1.2 Dialog mode

In dialog mode the dialog sequence is (user input underlined):

```
Diskname: :disk1
```

8. Config - Output Hardware Configuration

This program outputs a description of the hardware configuration.

8.1 Description

The configuration output is the logical configuration, that is, a comprehensive list showing which unit the running Unirex operating system may address.

The list shows the device names for terminals and printers. Furthermore, the list for each disk in the system shows devicename, disktype and size. Finally, the amount of memory associated to each cpu.

Config takes no parameters.

9. Copy - Copy Iounits

Copy is used to copy the contents of one iounit to another.

9.1 Description

The primary use of copy is to copy the contents of one file to another. Copy cannot copy a directory, but using wildcard characters, e.g. ':disk1/*', every file in a directory will be copied (also see diskcopy and wback).

However, it is possible to copy to/from other iounits than files, e.g. terminals, printers and boxes.

Copy takes as many as 5 parameters, some of which make sense only in special cases. The 5 parameters are: Source iounit, destination iounit, size of destination unit, copy mode, and permission to overwrite existing destination unit.

The iounit names of the source unit and the destination unit must be given.

Size of destination unit is relevant only if the destination unit is a non-existing file or box.

Copy mode and overwrite permission are relevant only if the destination unit is a file.

Irrelevant parameters are ignored by copy.

Size of destination unit is by default taken to be the same as the size of the source unit. However, some units are sizeless, e.g. terminals, and in these cases the default size of the destination unit is 2560 bytes for a file and 256 bytes for a box.

Copy mode may be 'o', 'c', or 'f', and the default value is 'o':

o: Ordinary copying: The entire file is copied, including not used filespace following a possible end of file mark.
c: Contiguous copying: The destination file is created with a size large enough to hold the entire source file. Using this mode, destination file size is ignored, and an existing destination file is deleted.
f: Formatted copying: The source file is considered a variabel length record file and are copied record by record until end of file is reached.

Overwrite permission may be 'y' or 'n' (default) for 'permission to overwrite an existing destination file is granted' or 'overwriting an existing destination must be accepted by the operator' respectively.

9.1.1 Parameter mode

In parameter mode the format is

```
        copy source destination (size) (mode) (overwrite)
```

and the keywords are:

source:	from
destination:	to
size:	size
mode:	mode
overwrite:	overwrite.

9.1.1.1 Examples

```
vox>copy from £i to :box/fnis
```

Copies input from the standard input device to box 'fnis'. If the box does not exist, it is created with a size of 256 bytes.

```
vox>copy from /hanoi-l to :disk1/hanoi-l size 5120 overwrite y
```

Copies the file hanoi-l on disk 0 to the file hanoi-l on disk1. If hanoi-l on disk 1 exists it is overwritten, otherwise it is created with a size of 5120 bytes.

9.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        Source unit:        £i
        Destination unit:   :box/fnis
        Size of dest.unit:  256
```

or

```
        Source unit:        /hanoi-l
        Destination unit:   :disk1/hanoi-l
        Size of dest.unit:  5120
        Copy mode:          o
        Overwrite:          y
```

10. Delete - Delete Iounits

This program removes a link in a directory to a file and/or deletes an iounit.

10.1 Description

Depending on the type of the iounit different requirements should be met, and different actions are taken:

Unifile file: Only privileged users may unlink a directory. To unlink a file the user must have write permission to the directory, but neither read nor write permission to the file itself is required. If the last link to a file is removed, the file is deleted.

Mikfile file: The user must have write permission to the file. The disk space occupied by a deleted file is not available for new files before the disk has been compressed (see compr).

Ordinary boxes: The user must have write permission to the box. The box is deleted even if it is not empty. If some process has the box open, deletion will be postponed until the box is closed. Note that empty boxes are automatically deleted when closed.

System boxes and common boxes: The user must be privileged. The box is deleted even if it is not empty. If some process has the box open, deletion will be postponed until the box is closed.

Delete may run in two modes: Auto deletion mode or confirm mode. Running in confirm mode the user must confirm the deletion of an iounit explicitly before the operation is performed.

Delete takes two parameters: the iounitname, and delete mode.

10.1.1 Parameter mode

In parameter mode the format is

```
        delete iounit (mode)
```

where mode is either 'y' for auto deletion mode or 'n' for confirm mode.

The keywords are:

```
        Iounit: unit
        Mode:   auto (default value: n)
```

10.1.1.1 Examples

```
vox>delete unit /hanoi-u auto y
```
Deletes the pascal p-code file hanoi on disk 0 without confirmation.

```
vox>delete unit :disk1/abc*
```
Delete on disk 1 all files which have the string 'abc' as the first part of their name. Ask for confirmation for each file.

10.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        Iounitname:   /hanoi-u
        Autodeletion: y
```
or
```
        Iounitname:   :disk1/abc*
        Autodeletion: n
        Delete unit :disk1/abcd ? y
        Delete unit :disk1/abc ? n
        Delete unit :disk1/abcdef ? y
```

11. Dir - Output Directory Contents

The dir program is used to list the contents of a directory or to list directory information about a file.

11.1 Description

The name of a Mikfile directory is the devicename of the logical disk, on which the directory resides.

The name of a Unifile main directory is the device name of the logical disk, on which the directory resides. The name of a subdirectory is the iounit name of the file containing the subdirectory.

Dir takes one parameter: the iounit name of a file or a directory. The default value is ':disk0'.

If the iounit name is the name of a file, dir outputs information about that file only.

The output format for the information output about a Unifile file has not been determined yet.

The information output about a Mikfile file is: The iounit name, filetype, size in bytes of base file, size in bytes of total file, record length in bytes, and the number of extents.

If the iounitname is the name of a directory, information about every not-deleted file in that directory is shown as described above. Finally, the size of unused diskspace, size of deleted files and the date of the last disk restore are output.

11.1.1 Parameter mode

In parameter mode the format is

```
        dir iounit
```

No keyword.

11.1.1.1 Example

```
vox>dir :disk1
```
Output a list of files in in the main directory of disk 1.

11.1.2 Dialog mode

In dialog mode the dialog sequence is (user input underlined):

```
Directory: :disk1
```

12. Diskcopy - Disk-to-disk Copy

This program is used to copy one entire disk to another.

12.1 Description

The two disks must be of the same type, that is, of the same size, and both disks must be unmounted.

Copying disk A to disk B also means to impress the filesystem of disk A on disk B. Thus disk B should be mounted with the same filesystem as disk A after copying.

Diskcopy, in contrast to wback, also copies the filesystem, and is less time consuming than copying with the copy program. In contrast to copy, diskcopy destroys the original contents of the destinationdisk.

Diskcopy takes three parameters: unit name of the source disk, unit name of the destinationdisk, and the size in bytes of the buffer used for copying.

12.1.1 Parameter mode

In parameter mode the format is

```
        diskcopy source destination (buffersize)
```

and the keywords are:

```
        source:      from
        destination: to
        buffersize:  size (default 64 Kb)
```

12.1.1.1 Example

```
vox>diskcopy from :disk1 to :disk2
```

Copy disk1 to disk2, using default buffersize.

12.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        Source disk:      :disk1
        Destination disk: :disk2
        Buffersize:       65536
```

13. Disksize - Output Size of Disk and Free Diskspace

This program outputs the type of a disk, the size of the disk, and the size of the free diskspace.

13.1 Description

The disk must be mounted prior to the execution of disksize. If the disk is a mikfile disk both the size of not used diskspace and the size occupied by deleted files are output.

Disksize takes one parameter: the iounitname of the disk.

13.1.1 Parameter mode

In parameter mode the format is

```
        disksize iounitname
```

No keyword.

13.1.1.1 Example

```
vox>disksize :disk0
```

Output size information about disk0.

13.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        Diskname: :disk0
```

14. Flink - Link a Unifile File to Directory

This program is used to create a link to a Unifile file from a directory.

14.1 Description

To create a link to a file from the directory, the user must have write access to the directory.

Having linked a file, the file is thereafter known by both the old and the new iounitname.

Flink cannot be used to copy a file and cannot establish multi volume links. This means that the device part of the existing and the new pathname must be the same.

Flink takes two parameters: old unitname, and new unitname.

14.1.1 Parameter mode

In parameter mode the format is:

```
        flink old_unitname  new_unitname
```

and the keywords are

```
        old_unitname: file
        new_unitname: as
```

14.1.1.1 Example

```
vox>flink from :disk3/alpha/beta as :disk3/gamma/delta
```

Create a path to the file alpha/beta on disk3 named gamma/delta. Write access of directory gamma is required.

14.1.2 Dialog mode

In dialog mode the dialog is (user input shown underlined):

```
        Old pathname: :disk3/alpha/beta
        New pathname: :disk3/gamma/delta
```

15. Init - Disk Initialization

This program initializes a disk, that is, creates a filesystem on a disk and leaves the disk otherwise empty.

15.1 Description

The disk must be unmounted prior to the execution of init.

A disk is initialized to a specific filesystem, and should after initialization be mounted to this filesystem.

Depending on the filesystem, init takes from 2 to 4 parameters. First init takes the filesystem name as a parameter, second init takes the iounit name of the disk as paramater. If the filesystem is Mikfile init in addition takes another 2 parameters: the disk label and the disk type.

Filetype should be 'u' for Unifile or 'm' for mikfile.

Disk label is an 8 character string, while disk type is a 5 character string. Neither disk label nor disk type are used by Mikfile, but required for compatibility to the Mikados file system.

15.1.1 Parameter mode

In parameter mode the format is

```
        init filesystem iounitname (disk_label) (disk_type)
```

and the keywords are

```
        filesystem: ftype (default value: m)
        iounitname: unit
        disk_label: label (default value: DDE)
        disk_type:  dtype (default value: MIKF)
```

15.1.1.1 Examples

```
vox>init ftype u unit :disk3
```

Delete all files on disk3, and impose an initially empty unifile filesystem on disk3.

```
vox>init unit :disk4
```

Delete all files on disk4, and impose an initially empty mikfile filesystem on disk4. Mark the disk to be of of type "MIKF" and give the disk the disk the label "DDE".

15.1.2 Dialog mode

In dialog mode the dialog is (user input underlined)

```
        Filesystem: u
        Diskname:   :disk3
```

or

```
        Filesystem: m
        Diskname:   :disk4
        Disk label: DDE
        Disk type:  MIKF
```

16. Install - Install Program

This program is used to install a program.

16.1 Description

Only privileged users may install programs.

Install takes two parameters: The iounitname of the file containing the program and the name to be assigned to the installed program.

No wildcards allowed.

16.1.1 Parameter mode

In parameter mode the format is

```
        install iounitname (programname)
```

and the keywords are:

```
        iounitname:  file
        programname: name (default value: first 8 characters of the last
component of the iounitname, exclusive filetype).
```

16.1.1.1 Examples

```
vox>install file :disk1/memdisp-l name display
```

The program contained in file :disk1/memdisp-l is installed under the name display.

```
vox>install file /comal-l
```

The program contained in file :disk0/comal-l is installed under the name comal.

16.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        Filename:    /comal-l
        Programname: comal
```

17. Mount - Mount Disk

This program is used to mount a disk, that is, to associate a disk with a filesystem.

17.1 Description

Mount takes one parameter: The iounit name of the disk.

17.1.1 Parameter mode

In parameter mode the format is

```
        Mount iounitname
```

No keyword.

17.1.1.1 Example

```
vox>mount :disk3
Disk 3 is mounted.
```

17.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        Disk name: :disk3
```

18. Prcsta - Process Status

This program is used to obtain a list of running processes.

18.1 Description

The Prcsta program writes a complete list of processes on the computer, which belong to the user. A privileged user will get a list not of his own processes only, but an entire list of all processes on the computer.

The processes are written in process number order. The name, number, user number, used memory, and status of each process are stated.

Be aware of the memory information is the total number of bytes used by the process - including memory which are shared with other processes.

The different process states are mentioned in "Unirex System Description".

The program terminates after writing the number of existing processes or when the user presses the attention key.

18.1.1 Example

vox>prcsta
An unpriviliged user will have the following list of processes on the terminal:

```
Unirex process status - version 22.04.1983
  no     name     user   memory   status
   12  copy       0103    13824  wait SIOC I/O
   14  prcsta     0103     9472  running
```

Number of existing processes : 15.

A priviliged user will have the following list of processes on the terminal:

```
Unirex process status - version 22.04.1983
  no     name     user   memory   status
    0  >inilog<      0        0  wait file I/O
    1  >dummy<       0        0  active
    2  mikfile0      0    77056  in off_sta
    3  mikfil-a      0    77056  wait box I/O
    4  mikfil-b      0    77056  wait box I/O
  5  mikfil-c      0     77056  wait box I/O
  6  mikfil-d      0     77056  wait box I/O
  7  mikfil-e      0     77056  wait box I/O
  8  logon0        0     13056  in off_sta
  9  logon1        0     13056  int.suspend
 10  dde        0103     15616  in off_sta
 11  dde        0201     15616  in off_sta
 12  copy       0103     13824  wait SIOC I/O
 13  inter      0201     66560  in off_sta
 14  prcsta     0103      9472  running

Number of existing processes : 15.
```

19. Remove - Remove Installed Program

This program is used to remove an installed program.

19.1 Description

Only privileged users may remove programs.

Remove takes one parameter: The name of the program to remove.

19.1.1 Parameter mode

In parameter mode the format is
```
        remove programname
```
No keyword.

19.1.1.1 Example

```
vox>remove comal
Program comal is removed.
```

19.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        Programname: comal
```

20. Rename - Rename iounit

This program renames a file.

20.1 Description

Rename takes two parameters: old unitname and new unit name.

The disk device for old and new unitname must be the same.

In the case of a unifile file rename requires write access to the new and old directory.

In the case of a mikfile file rename requires write access to the file.

20.1.1 Parameter mode

```
In parameter mode the format is
        rename old_unitname new_unitname
and the keywords are
        old_unitname: from
        new_unitname: to
```

20.1.1.1 Example

```
vox>rename from /alpha-l to /beta-l
The file alpha-l on disk0 is renamed to beta-l.
```

20.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        Old filename: /alpha-l
        New filename: /beta-l
```

21. Setsioc - Set SIOC Characteristics

The setsioc program can be used to set

1) The terminal/printer type
2) The baud rate, number of stop bits, number of data bits, and parity used in the transmission
3) The characters that represent attention, x-on, and x-off.

21.1 Terminal/Printer type

The terminal/printer type is a number in the range 0 to 7, which chooses from a set of internally stores terminal/printer types.

The user may chooose not to alter the type.

21.2 UART Characteristics

The user may choose to change none or all of the UART characteristics. If the user chooses to change one of the UART characteristics (baud rate, data bits, stop bits, or parity) all the values will be set, either to the value specified by the user, or to the dafault values, which are 9600 baud, 7 data bits, 2 stop bits, and even parity.

21.3 Attention, X-on, X-off

The characters for attention, x-on, and x-off, are reprsented by a mask and a value. Akey pressed on the keyboard is logically and'ed with the mask, whereupon it is checked if it matches attention, x-on, or x-off.

The user may choose to change none or all of these specifications. If the user chooses to change either the mask or attention, x-on, or x-off, all the values will be set, either to the value specified by the user, or to the default values, which are mask = 7f (hexadecimal), x-on = ctrl/q, x-off = ctrl/s, and attention = escape.

21.4 Parameter mode

In parameter mode the following keywords apply:

unit	followed by a terminal or printer specification.
type	followed by an integer between 0 and 7.
baud	followed by an integer that is a legal baud rate.

data	followed by an integer between 5 and 8.
stop	followed by 1, 1.5, or 2.
parity	followed by even, odd, or none.
mask	followed by a 2-digit hexadecimal number.
x-on	followed by a 2-digit hexadecimal number.
x-off	followed by a 2-digit hexadecimal number.
att	followed by a 2-digit hexadecimal number.

If the type is specified the type will be changed.

If either baud, data, stop, or parity is specified, they will all be changed.

If either mask, x-on, x-off, or att is specified, they will all be changed.

21.5 Dialog mode

If no parameters are given, the program will run in dialog mode. The prompts are self-explanatory; it should however be noted that the suer may choose not to alter a set a characteristics by entering ctrl/c when the program informs of this possibility.

22. Settime - Set Current Date and Time

Settime sets the systems date and time.

22.1 Description

Settime takes two parameters: date and time.

Note that the date should be given in the european standard format dd.mm.yy and that time should be given as hh:mm:ss.

22.1.1 Parameter mode

In parameter mode the format is

```
        settime date time
```

and the keywords are

```
        date: date (default value: current date)
        time: time (default value: current time)
```

22.1.1.1 Example

```
vox>settime date 11.04.83 time 15:21:30
```

22.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        New date (dd.mm.yy): 11.04.83
        New time (hh:mm:ss): 15:21:30
```

23. Time - Output Date and Time

Time outputs the current date and time.

23.1 Description

Time takes no parameters.

The date and time is presented in format:

```
        day_of_the_week day_number name_of_the_month year hh:mm:ss
e.g.
        monday  11 apr 1983   15:21:50
```

24. Unmount - Unmount Disk

This program is used to unmount a disk, that is, to disassociate the disk from a filesystem.

24.1 Description

If there is open files on the disk unmount is postponed until there is no more open files on the disk.

Since unmounting may mean synchronization of some filesystem parameters, a disk should be unmounted before removed from the disk drive.

Unmount takes one parameter: The iounit name of the disk.

24.1.1 Parameter mode

In parameter mode the format is

```
        unmount iounitname
```

No keyword.

24.1.1.1 Example

```
vox>unmount :disk3
Disk 3 is unmounted.
```

24.1.2 Dialog mode

In dialog mode the dialog is (user input underlined):

```
        Disk name: :disk3
```

25. Version - Output Unirex Version

Outputs the version identification of the Unirex operating system.

25.1 Description

The Unirex operating system is identified by a version date.

Version takes no parameters.

```
The version date is presented like
        Unirex Operating System, version 11 apr 1983
using the format
        day_of_the_week day_number name_of_the_month year hh:mm:ss
```

26. Vox - Operator Communication

Vox is the standard operator communication program running under Unirex. Vox enables the terminal operator to start the execution of named programs, pass parameters to the started programs and define the standard environment for the started programs.

Vox accepts input from a terminal or from another iounit. If vox takes input from a terminal, vox is said to be running in terminal mode, otherwise vox is said to be running in monitor mode.

Running in terminal mode vox prompts 'vox>' or 'vox>>' to show that commands may be entered. The prompt 'vox>>' is used when the user is a privileged user. Thus the prompt emphasizes the operator status, and reminds priviliged users to be carefull.

26.1 Commands

Vox accepts 3 commands: Stop, set current unit prefix and start program.

26.1.1 The Stop Command

The format for the stop command is

```
        .stop
```

This command terminates vox.

26.1.2 Set Current Unit Prefix

The format for set current unit prefix is

```
        .cup current-unit-prefix
```

where current-unit-prefix is a string of printable ascii characters exclusive spaces.

This command sets the current unit prefix for vox and for any process started by vox.

The current unit prefix is reset to the empty string by

```
        .cup
```

Also see about current unit prefix in Unirex System Description, 7 march 1983.

26.1.3 Start program

This command is used to start a program, either as an off-spring process or as a produced process.

The format is

```
        (mode)unitname (program-parameter) (standard-unit-list)
```

where the components have the following semantic:

mode Mode may be omitted, @ or ^. If mode is omitted the program is spawned, if @ the program is produced, and if ^ the program is gemated.

unitname is the iounitname of the file containing the program. A file type must not be given, type l is implied. If the first character of unitname is ':', '/' or '!', e.g. ':disk0/inter', '/inter' or '!mcu1:disk1/inter', the program is loaded from the named disk.

In any other case vox first tries to start an installed program, then to load the program using current unit prefix as prefix to unitname, and last to load the program from disk0.

If the program can not be found vox search for a file of type u (pascal p-code file) with the given unitname. If the first character of unitname is ':', '/' or '!', the file is searched on the named disk is only. Otherwise vox first search for the program using current unit prefix as a prefix to the unitname and then on disk0. If a type u file is found, inter is started using unitname as parameter.

If a type u file is not found vox searches for a file of type q (binary comal program) with the given unitname. The search is analog to the search for a type u file. If a type q file is found, comal is started in verify mode using unitname as parameter.

program-parameter is a string of printable ascii characters inclusive spaces. The program-parameter is passed to the started program as parameter. If inter or comal are started automatically the program-parameter is appended to unitname and separated from unitname by one space character.

The number of characters is the resulting parameter must not exceed 70.

Spaces are not significant in the sense that a number of consecutive spaces are passed as one space. However, spaces enclosed in apostrophes are passed as spaces.

Program-parameter must not contain any of keywords listed below. The first keyword starts the standard-unit-list.

standard-unit-list indicates which iounits the started program should use for standard units. When no standard-unit-list is given the started program inherits the standard iounits from vox, that is STDIN, STDOUT, STDERR and STDLIST. Otherwise the elements in standard-unit-list redefines the standard units.

An element in standard-unit-list combines a keyword and a unitname.

The keywords are:

input:	The unit is opened and the ioud is passed as STDIN.
output:	The unit is opened and the ioud is passes as STDOUT.
aoutput:	As output. The unit is positioned to the first free byte.
error:	The unit is opened and the ioud is passed as STDERR.
aerror:	As error. The unit is positioned to the first free byte.
list:	The unit is opened and the ioud is passed as STDLIST.
alist:	As list. The unit is positioned to the first free byte.

The elements in standard-unit-list may appear in any order.

Wildcard characters are not allowed.

26.2 Multiple lines commands

A vox command may be read from several lines. If the input string terminates by '++', vox automatically asks for more input before the command is interpreted.

A keyword and the associated unitname must appear in the same line.

The stringterminator '++' has the same meaning as a space, that is the lines:

```
/inter++
/hanoi
```

are totally equivalent to

```
/inter /hanoi.
```

26.3 Recall function

Pressing the restore function key allows the terminal operator to edit the last input line. In case this was part of a multiple line command the original line is replaced by the edited line.

26.4 Process death

When an offspring process dies information about this is reported to vox. Vox reports this event by sending a message to STDOUT unless the death was reported to be normal. Normal death is indicated by a completion code 0.

26.5 Monitor mode

Vox enters monitor mode if vox is started with a standard unit list specifying that input is taken from an iounit which is not a terminal.

In this case all input is read from the specified unit, and vox is terminated by a stop command or by reading an end-of-file mark.

Every command is echoed on the standard output unit. This unit may have been set to the null device, thus suppressing the output from vox.

Programs started from a vox program running in monitor take input from the same unit as vox, unless otherwise specified.

26.6 Examples

```
vox>/pasuni /hanoi list /sourlist-k
```
Load and start pasuni from disk0. The parameter to pasuni is '/hanoi'. For STDIN, STDOUT and STDERR the corresponding vox standard units are used, while output to STDLIST is written in the file '/sourlist-k'.

```
vox>vox input /monfil-k output :null
```
Start vox in monitor mode. The started vox takes input from the file '/monfil-k' on disk0, and echoes commands on the null device.

27. Wback - Disk Back-up

Wback is used to produce a back-up copy of a disk or to restore a disk from a back-up.

27.1 Description

Any disk may be used to hold a back-up of a disk. It should be noted, that a streamer tape is conceptually a disk, thus a back-up disk may as well be a streamer tape.

However, if the back-up disk is a streamer tape, the tape capacity must be big enough to contain the entire source disk. If the back-up disk is actually a disk the wback program automatically instructs the operator to insert a new back-up disk when the current back-up disk is full.

Both the source and destination disk must be unmounted prior to a back-up or a disk restore.

The produced back-up is a dump of the source disk, which means, that single files cannot be extracted from a back-up. This property may be achieved using the copy or diskcopy program.

Before restoring a disk wback checks that the disk to be restored has the same type as the backed-up disk.

The date of back-up is written on the restored disk and may be shown to the user using the utility dir.

Wback takes four parameters: source medium, destination medium, a mode para-
meter, and size in bytes of buffer used for copying.
Source medium is the device name of the disk to read, either the device name of the disk to be backed-up or the device name of the disk, on which the back-up resides (restoring). Mode is either 'b' for backup or 'r' for restore.

27.1.1 Parameter mode

In parameter mode the format is:

```
        wback  source destination mode (size)
```

and the keywords are

```
        source:      from
```

```
destination: to
mode:        mode
buffersize:  size (default value 32768)
```

27.1.1.12 Example

```
vox>wback from :disk0 to :disk5
```

Disk 0 is dumped on disk 5 after a user acceptance.

27.1.2 Dialog mode

In dialog mode the dialog is (user input shown underlined):

```
Source disk:       :disk0
Destination disk:  :disk5
Buffersize:        32768
```

In dialog mode the user accept of the copying is always required.