

NAME

cat – phototypesetter interface

DESCRIPTION

Cat provides the interface to a Graphic Systems C/A/T phototypesetter. Bytes written on the file specify font, size, and other control information as well as the characters to be flashed. The coding will not be described here.

Only one process may have this file open at a time. It is write-only.

FILES

/dev/cat

SEE ALSO

troff (I), Graphic Systems specification (available on request)

BUGS

NAME

da – voice response unit

DESCRIPTION

Bytes written on this file control a Cognitronics optical drum voice response unit which can generate up to 31 fixed half-second utterances. Bytes read correspond to Touch-Tone® signals received via a 403 dataset.

The specifics of the interface will not be described. Consult M. E. Lesk for more information.

FILES

/dev/da

BUGS

NAME

dc – DC-11 communications interface

DESCRIPTION

The special files /dev/tty0, /dev/tty1, ... refer to the DC11 asynchronous communications interfaces. At the moment there are 12 of them, but the number is subject to change.

When one of these files is opened, it causes the process to wait until a connection is established. In practice user's programs seldom open these files; they are opened by *init* and become a user's input and output file. The very first typewriter file open in a process becomes the *control typewriter* for that process. The control typewriter plays a special role in handling quit or interrupt signals, as discussed below. The control typewriter is inherited by a child process during a *fork*.

A terminal associated with one of these files ordinarily operates in full-duplex mode. Characters may be typed at any time, even while output is occurring, and are only lost when the system's character input buffers become completely choked, which is rare, or when the user has accumulated the maximum allowed number of input characters which have not yet been read by some program. Currently this limit is 256 characters. When the input limit is reached all the saved characters are thrown away without notice.

When first opened, the interface mode is 150 baud; either parity accepted; 10 bits/character (one stop bit); and newline action character. The system delays transmission after sending certain function characters. Delays for horizontal tab, newline, and form feed are calculated for the Teletype Model 37; the delay for carriage return is calculated for the GE TermiNet 300. Most of these operating states can be changed by using the system call *stty(II)*. In particular the following hardware states are program settable independently for input and output (see DC11 manual): 134.5, 150, 300, or 1200 baud; one or two stop bits on output; and 5, 6, 7, or 8 data bits/character. In addition, the following software modes can be invoked: acceptance of even parity, odd parity, or both; a raw mode in which all characters may be read one at a time; a carriage return (CR) mode in which CR is mapped into newline on input and either CR or line feed (LF) cause echoing of the sequence LF-CR; mapping of upper case letters into lower case; suppression of echoing; suppression of delays after function characters; and the printing of tabs as spaces. See *getty(VII)* for the way that terminal speed and type are detected.

Normally, typewriter input is processed in units of lines. This means that a program attempting to read will be suspended until an entire line has been typed. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not however necessary to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.

During input, erase and kill processing is normally done. The character '#' erases the last character typed, except that it will not erase beyond the beginning of a line or an EOT. The character '@' kills the entire line up to the point where it was typed, but not beyond an EOT. Both these characters operate on a keystroke basis independently of any backspacing or tabbing that may have been done. Either '@' or '#' may be entered literally by preceding it by '\'; the erase or kill character remains, but the '\' disappears.

In upper-case mode, all upper-case letters are mapped into the corresponding lower-case letter. The upper-case letter may be generated by preceding it by '\'. In addition, the following escape sequences are generated on output and accepted on input:

for	use
\	\\
	\\
~	\\~
{	\\{
}	\\}

It is possible to use raw mode in which the program reading is awakened on each character. In raw mode, no erase or kill processing is done; and the EOT, quit and interrupt characters are not treated specially.

The ASCII EOT character may be used to generate an end of file from a typewriter. When an EOT is received, all the characters waiting to be read are immediately passed to the program, without waiting for a new-line. Thus if there are no characters waiting, which is to say the EOT occurred at the beginning of a line, zero characters will be passed back, and this is the standard end-of-file signal. The EOT is not passed on except in raw mode.

When the carrier signal from the dataset drops (usually because the user has hung up his terminal) a *hangup* signal is sent to all processes with the typewriter as control typewriter. Unless other arrangements have been made, this signal causes the processes to terminate. If the hangup signal is ignored, any read returns with an end-of-file indication. Thus programs which read a typewriter and test for end-of-file on their input can terminate appropriately when hung up on.

Two characters have a special meaning when typed. The ASCII DEL character (sometimes called 'rubout') is not passed to a program but generates an *interrupt* signal which is sent to all processes with the associated control typewriter. Normally each such process is forced to terminate, but arrangements may be made either to ignore the signal or to receive a simulated trap to an agreed-upon location. See signal (II).

The ASCII character FS generates the *quit* signal. Its treatment is identical to the interrupt signal except that unless a receiving process has made other arrangements it will not only be terminated but a core image file will be generated. See signal (II).

Output is prosaic compared to input. When one or more characters are written, they are actually transmitted to the terminal as soon as previously-written characters have finished typing. Input characters are echoed by putting them in the output queue as they arrive. When a process produces characters more rapidly than they can be typed, it will be suspended when its output queue exceeds some limit. When the queue has drained down to some threshold the program is resumed. Even-parity is always generated on output. The EOT character is not transmitted (except in raw mode) to prevent terminals which respond to it from hanging up.

FILES

/dev/tty[01234567abcd] 113B Dataphones

SEE ALSO

kl (IV), getty (VII), stty (I, II), gtty (I, II), signal (II)

BUGS

NAME

dn – dn11 ACU interface

DESCRIPTION

The *dn?* files are write-only. The permissible codes are:

0-9 dial 0-9
: dial *
; dial #
= end-of-number

The entire telephone number must be presented in a single *write* system call.

It is recommended that an end-of-number code be given even though only one of the ACU's (113C) actually requires it.

FILES

/dev/dn0 connected to 801 with dp0
/dev/dn1 connected to 113C with ttyc
/dev/dn2 not currently connected

SEE ALSO

dp(IV), dc(IV), write(II)

BUGS

It needs a delay character to handle second dial tone.

NAME

dp – dp11 201 data-phone interface

DESCRIPTION

The *dp0* file is a 201 data-phone interface. *Read* and *write* calls to *dp0* are limited to a maximum of 512 bytes. Each write call is sent as a single record. Seven bits from each byte are written along with an eighth odd parity bit. The sync must be user supplied. Each read call returns characters received from a single record. Seven bits are returned unaltered; the eighth bit is set if the byte was not received in odd parity. A 10 second time out is set and a zero-byte record is returned if nothing is received in that time.

FILES

/dev/dp0

SEE ALSO

dn(IV), gerts(III)

BUGS

NAME

kl – KL-11/TTY-33 console typewriter

DESCRIPTION

Tty (as distinct from *tty?*) refers to the console typewriter hard-wired to the PDP-11 via a KL-11 interface. The disciplines involved in dealing with *tty* are identical to those for *tty?* and section DC(I) should be consulted. The following differences are salient:

The system calls *stty* and *gtty* apply, and the bits in the mode word have the same meanings, but the speed-select word is ignored. The quit signal is generated by the key marked 'alt mode.'

By appropriate console switch settings, it is possible to cause UNIX to come up as a single-user system with I/O on this device.

FILES

/dev/tty
/dev/tty8 synonym for /dev/tty
/dev/tty9 second console

SEE ALSO

dc(IV), init(VII)

BUGS

NAME

mem — core memory

DESCRIPTION

Mem is a special file that is an image of the core memory of the computer. It may be used, for example, to examine, and even to patch the system using the debugger.

A memory address is an 18-bit quantity which is used directly as a UNIBUS address. References to non-existent locations cause errors to be returned.

Examining and patching device registers is likely to lead to unexpected results when read-only or write-only bits are present.

FILES

/dev/mem

BUGS

There should be another *mem* file that looks at core using the system's address map.

NAME

pc – PC-11 paper tape reader/punch

DESCRIPTION

Ppt refers to the PC-11 paper tape reader or punch, depending on whether it is read or written.

When *ppt* is opened for writing, a 100-character leader is punched. Thereafter each byte written is punched on the tape. No editing of the characters is performed. When the file is closed, a 100-character trailer is punched.

When *ppt* is opened for reading, the process waits until tape is placed in the reader and the reader is on-line. Then requests to read cause the characters read to be passed back to the program, again without any editing. This means that several null leader characters will usually appear at the beginning of the file. Likewise several nulls are likely to appear at the end. End-of-file is generated when the tape runs out.

Seek calls for this file are meaningless.

FILES

/dev/ppt

BUGS

NAME

rf – RF11/RS11 fixed-head disk file

DESCRIPTION

This file refers to the concatenation of all RS-11 disks.

Each disk contains 1024 256-word blocks. The length of the combined RF file is $1024 \times (\text{minor} + 1)$ blocks. That is minor device zero is 1024 blocks long; minor device one is 2048, etc.

FILES

/dev/rf0

BUGS

NAME

rk – RK-11/RK03 (or RK05) disk

DESCRIPTION

Rk? refers to an entire RK03 disk as a single sequentially-addressed file. Its 256-word blocks are numbered 0 to 4871.

Drive numbers (minor devices) of eight and greater are treated specially. Drive 8+*x* is the *x*+1 way interleaving of devices rk0 to rk*x*. Thus blocks on rk10 are distributed alternately among rk0, rk1, and rk2.

FILES

/dev/rk?

BUGS

Care should be taken in using the interleaved files. First, the same drive should not be accessed simultaneously using the ordinary name and as part of an interleaved file, because the same physical blocks have in effect two different names; this fools the system's buffering strategy. Second, the combined files cannot be used for swapping.

NAME

rp – RP-11/RP03 moving-head disk

DESCRIPTION

The files *rp0* ... *rp7* refer to sections of RP disk drive 0. The files *rp8* ... *rp15* refer to drive 1 etc. This is done since the size of a full RP drive is 81200 blocks and internally the system is only capable of addressing 65536 blocks. Also since the disk is so large, this allows it to be broken up into more manageable pieces.

The origin and size of the pseudo-disks on each drive are as follows:

disk	start	length
0	0	40600
1	40600	40600
2	0	3200
3	3200	39000
4	42200	39000
5-7	unassigned	

FILES

/dev/rp?

BUGS

NAME

tc – TC-11/TU56 DECtape

DESCRIPTION

The files *tap0* ... *tap7* refer to the TC-11/TU56 DECtape drives 0 to 7.

The 256-word blocks on a standard DECtape are numbered 0 to 577.

FILES

/dev/tap?

SEE ALSO

tp(I)

BUGS

Since reading is synchronous, only one block is picked up per tape reverse.

NAME

tiu – Spider interface

DESCRIPTION

Spider is a fast digital switching network. *Tiu* is a directory which contains files each referring to a Spider control or data channel. The file `/dev/tiu/dn` refers to data channel *n*, likewise `/dev/tiu/cn` refers to control channel *n*.

The precise nature of the UNIX interface has not been defined yet.

FILES

`/dev/tiu/d?`, `/dev/tiu/c?`

BUGS

NAME

tm – TM-11/TU-10 magtape interface

DESCRIPTION

The files *mt0*, ..., *mt7* refer to the DEC TU10/TM11 magtape. When opened for reading or writing, the magtape is rewound. A tape consists of a series of 512 byte records terminated by an end-of-file. When the magtape is closed after writing, an end-of-file is written.

The magtape can only be opened once at any instant.

FILES

/dev/mt?

SEE ALSO

tp(I)

BUGS

If you hit the EOF mark or get other non-data errors it refuses to do anything more until closed. There has to be some reasonable way to deal with multi-file tapes.

NAME

vs – voice synthesizer interface

DESCRIPTION

Bytes written on *vs* drive a Federal Screw Works Votrax® voice synthesizer. The upper two bits encode an inflection, the other 6 specify a phoneme. The code is given in section *vs* (VII).

Touch-Tone® signals sent by a caller will be picked up during a *read* as the ASCII characters {0123456789#*}.

FILES

/dev/vs

SEE ALSO

speak (I), *vs* (VII)

BUGS

NAME

vt – 11/20 (vt01) interface

DESCRIPTION

The file *vt0* provides the interface to a PDP 11/20 which runs a VT01A-controlled Tektronix 611 storage display. The inter-computer interface is a pair of DR-11C word interfaces.

Although the display has essentially only two commands, namely “erase screen” and “display point”, the 11/20 program will draw points, lines, and arcs, and print text on the screen. The 11/20 can also type information on the attached 33 TTY.

This special file operates in two basic modes. If the first byte written of the file cannot be interpreted as one of the codes discussed below, the rest of the transmitted information is assumed to ASCII and written on the screen. The screen has 33 lines (1/2 a standard page). The file simulates a 37 TTY: the control characters NL, CR, BS, and TAB are interpreted correctly. It also interprets the usual escape sequences for forward and reverse half-line motion and for full-line reverse. Greek is not available yet. Normally, when the screen is full (i.e. the 34th line is started) the screen is erased before starting a new page. To allow perusal of the displayed text, it is usual to assert bit 0 of the console switches. This causes the program to pause before erasing until this bit is lowered.

If the first byte written is recognizable, the display runs in graphic mode. In this case bytes written on the file are interpreted as display commands. Each command consists of a single byte usually followed by parameter bytes. Often the parameter bytes represent points in the plotting area. Each point coordinate consists of 2 bytes interpreted as a 2's complement 16-bit number. The plotting area itself measures $(\pm 03777) \times (\pm 03777)$ (numbers in octal); that is, 12 bits of precision. Attempts to plot points outside the screen limits are ignored.

The graphic commands follow.

order (1); 1 parameter byte

The parameter indicates a subcommand, possibly followed by subparameter bytes, as follows:

erase (1)

The screen is erased. The program will wait until bit 0 of the console switches is down.

label (3); several subparameter bytes

The following bytes up to a null byte are printed as ASCII text on the screen. The origin of the text is the last previous point plotted; or the upper left hand of the screen if there were none.

point (2); 4 parameter bytes

The 4 parameter bytes are taken as a pair of coordinates representing a point to be plotted.

line (3); 8 parameter bytes

The parameter bytes are taken as 2 pairs of coordinates representing the ends of a line segment which is plotted. Only the portion lying within the screen is displayed.

frame (4); 1 parameter byte

The parameter byte is taken as a number of sixtieths of a second; an externally-available lead is asserted for that time. Typically the lead is connected to an automatic camera which advances its film and opens the shutter for the specified time.

circle (5); 6 parameter bytes

The parameter bytes are taken as a coordinate pair representing the origin, and a word representing the radius of a circle. That portion of the circle which lies within the screen is plotted.

arc (6); 12 parameter bytes

The first 4 parameter bytes are taken to be a coordinate-pair representing the center of a circle. The next 4 represent a coordinate-pair specifying a point on this circle.

The last 4 should represent another point on the circle. An arc is drawn counter-clockwise from the first circle point to the second. If the two points are the same, the whole circle is drawn. For the second point, only the smaller in magnitude of its two coordinates is significant; the other is used only to find the quadrant of the end of the arc. In any event only points within the screen limits are plotted.

dot-line (7); at least 6 parameter bytes

The first 4 parameter bytes are taken as a coordinate-pair representing the origin of a dot-line. The next byte is taken as a signed x-increment. The next byte is an unsigned word-count, with '0' meaning '256'. The indicated number of words is picked up. For each bit in each word a point is plotted which is visible if the bit is '1', invisible if not. High-order bits are plotted first. Each successive point (or non-point) is offset rightward by the given x-increment.

Asserting bit 3 of the console switches causes the display processor to throw away everything written on it. This sometimes helps if the display seems to be hung up.

FILES

/dev/vt0

BUGS