NAME mem -- core memory

SYNOPSIS

DESCRIPTION _mem_ maps the core memory of the computer into a file. It may be used, for example, to examine, and even to patch the system using the debugger.

Mem is a byte-oriented file; its bytes are numbered 0 to 65,535.

FILES

SEE ALSO

DIAGNOSTICS

BUGS If a location not corresponding to implemented memory is read or written, the system will incur a bus-error trap and, in panic, will reboot itself.

OWNER ken, dmr

NAME ppt -- punched paper tape

SYNO?S IS

DESCRIPTION ppt refers to the paper tape reader or punch, depending on whether it is read or written.

When 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 characters will usually appear at the beginning of the file; they correspond to the tape leader. 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 and are effectively ignored (however, the read/write pointers are maintained and an arbitrary sequence of reads or writes intermixed with seeks will give apparently correct results when checked with tell).

FILES

SEE ALSO lbppt, dbppt, bppt format

DIAGNOSTICS

BUGS Previously, there were separate special files for ASCII tape (which caused null characters to be suppressed) and binary tape (which used a blocked format with checksums). These notions were conceptually quite attractive, but they were discarded to save space in the system.

OWNER ken, dmr

NAME rf0 -- RF11—RS11 fixed—head disk file

SYNOPSIS

DESCRIPTION This file refers to the entire RF disk. It may be either read or written, although writing is inherently very dangerous, since a file system resides there.

The disk contains 1024 256—word blocks, numbered 0 to 1023. Like the other block—structured devices (tape, RK disk) this file is addressed in blocks, not bytes. This has two consequences: *seek* calls refer to block numbers, not byte numbers; and sequential reading or writing always advance the read or write pointer by at least one block. Thus successive reads of 10 characters from this file actually read the first 10 characters from successive blocks.

FILES

SEE ALSO /dev/tap0, /dev/rk0

DIAGNOSTICS

BUGS The fact that this device is addressed in terms of blocks, not bytes, is extremely unfortunate. It is due entirely to the fact that read and write pointers (and consequently the arguments to seek and *tell*) are single—precision numbers. This really has to be changed but unfortunately the repercussions are serious.

OWNER ken, dmr

NAME rk0 -- RK03 (or RK05) disk

SYNOPSIS

DESCRIPTION rk0 refers to the entire RK03 disk as a single sequentially-addressed file. Its 256-word blocks are numbered 0 to 4871. Like the RF disk and the tape files, its addressing is block-oriented. Consult the /dev/rf0 section.

FILES

SEE ALSO /dev/rf0, /dev/tap0

DIAGNOSTICS

BUGS See /dev/rf0

OWNER ken, dmr

NAME tap0 ... tap7

SYNOPSIS

DESCRIPTION These files refer to DECtape drives 0 to 7. Since the logical drive number can be manually set, all eight files exist even though at present there are only two physical drives.

The 256—word blocks on a standard DECtape are numbered 0 to 577. However, the system makes no assumption about this number; a block can be read or written if it exists on the tape and not otherwise. An error is returned if a transaction is attempted for a block which does not exist.

Like the RK and RF special files, addressing on the tape files is block—oriented. See the RF0 section.

FILES

SEE ALSO /dev/rf0, /dev/rk0

DIAGNOSTICS

BUGS see /dev/rf0

OWNER ken, dmr

NAME tty -- console typewriter

SYNOPSIS

DESCRIPTION _tty_ (as distinct from tty0 ..., _tty5_) refers to the console typewriter hard-wired to the PDP-11. Most of the time it is turned off and so has little general use.

Generally, the disciplines involved in dealing with _tty_ are similar to those for _tty0_ ... and the appropriate section should be consulted. The following differences are salient:

The system calls _stty_ and _gtty_ do not apply to this device. It cannot be placed in raw mode; on input, upper case letters are always mapped into lower case letters; a carriage return is echoed when a line-feed is typed.

The quit character is not FS (as with tty0...) but is generated by the key labelled "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

SEE ALSO /dev/tty0, ...; init

DIAGNOSTICS

BUGS

OWNER ken, dmr

NAME tty0 ... tty5 -- communications interfaces

SYNOPSIS

DESCRIPTION

These files refer to DC11 asynchronous communications interfaces. At the moment there are six of them, but the number is subject to change. Names for up to four others will be constructed by an obvious algorithm.

When one of these files is opened, it causes the process to wait until a connection is established. (In practice, however, user's programs seldom open these files; they are opened by init and become a user's standard 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 the 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 very rare.

When first opened, the interface expects the terminal to use 15 odd—parity, 10—bit ASCII characters per second and to have the new—line function. Finally, the system calculates delays after sending the code for certain functions (e.g., new—line, tab) on the assumption that the terminal is a Teletype model 37. All this is merely a long way of saying that the system expects to be used by a TTY 37. However, most of these assumptions can be changed by a special system call: in particular, the expected parity can be changed; the speed, character size, and stop bits can be changed (speeds available are 134.5, 150, 300, 1200 baud; see the DC11 manual); the new—line function can be simulated by a combination of the carriage—return and line—feed functions; carriage return can be translated into new—line on input; upper case letters can be mapped into lower case letters; echoing can be turned off so the terminal operates in half duplex. See the system call _stty_. (Also see _init_ for the way 300—baud terminals are detected.)

Normally, a typewriter operates 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.

The 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.

When the carrier signal from the dataset drops (usually because the user has hung up his terminal) any read returns with an end–of–file indication. Thus programs which read a typewriter and are sensitive to end–of–file on their inputs (which all programs should be) will terminate appropriately when hung up on.

Two characters have a special meaning when typed. The ASCII DEL character (sometimes called rubout") is the interrupt signal. When this character is received from a given typewriter, a search is made for all processes which have this typewriter as their control typewriter, and which have not informed the system that they wish to ignore interrupts. If there is more than one such process, one of these is selected, for practical purposes at random. Then either the process is forced to exit or a trap is simulated to an agreed–upon location in the process. See intr for more information.

The ASCII character FS is the quit signal. Its treatment is identical to the interrupt signal except that unless the receiving process has made other arrangements it will not only be terminated but a core image file will be written. (See quit for more 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 EOF. The character "@" kills the entire line up to the point where it was typed, but not beyond an EOF. 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.

It is also possible (again by sys stty) to put the typewriter into raw mode. In this mode, the program reading is wakened on each character, and when a program reads, it waits only until at least one character has been typed. In raw mode, no erase or kill processing is done; and the EOT, quit and interrupt characters are not treated specially.

Output is prosaic compared to input. It should be noted, however, that when one or more characters are written, they are actually transmitted to the terminal as soon as previously–written characters have finished typing. When a program produces characters too rapidly to be typed, as is very common, it may be suspended for a time.

Odd parity is always generated on output, except that the characters EOT and NAK have the wrong parity. Thus the 37 TTY will not hang up (EOT) or lock its keyboard (NAK) if a program accidentally prints these characters.

FILES

SEE ALSO tty

DIAGNOSTICS

BUGS As has been suggested, UNIX has a heavy predisposition towards 37 Teletype terminals. However, it is quite possible to use 300–baud terminals such as the GE TermiNet 300. (See init for the procedure.) The main difficulty in practice is 37–oriented delay calculations.

Terminals such as the IBM 2741 would theoretically be very desirable but there are many difficulties related to its inadequate and non–ASCII character sets (the 2741 has two, count 'em) and the inherently half–duplex nature of the terminal. It is possible to produce output on a 2741; cf type.

OWNER ken, dmr