

## INTRODUCTION TO SYSTEM CALLS

Section II of this manual lists all the entries into the system. In most cases two calling sequences are specified, one of which is usable from assembly language, and the other from C. Most of these calls have an error return. From assembly language an erroneous call is always indicated by turning on the c-bit of the condition codes. The presence of an error is most easily tested by the instructions *bes* and *bec* ("branch on error set (or clear)"). These are synonyms for the *bcs* and *bcc* instructions.

From C, an error condition is indicated by an otherwise impossible returned value. Almost always this is -1; the individual sections specify the details.

In both cases an error number is also available. In assembly language, this number is returned in r0 on erroneous calls. From C, the external variable *errno* is set to the error number. *Errno* is not cleared on successful calls, so it should be tested only after an error has occurred. There is a table of messages associated with each error, and a routine for printing the message. See *perorr* (III).

The possible error numbers are not recited with each writeup in section II, since many errors are possible for most of the calls. Here is a list of the error numbers, their names inside the system (for the benefit of system-readers), and the messages available using *perorr*. A short explanation is also provided.

- |   |         |   |
|---|---------|---|
| 0 | -       | (unused)  |
| 1 | EPERM   | Not owner and not super-user<br>Typically this error indicates an attempt to modify a file in some way forbidden except to its owner. It is also returned for attempts by ordinary users to do things allowed only to the super-user.   |
| 2 | ENOENT  | No such file or directory<br>This error occurs when a file name is specified and the file should exist but doesn't, or when one of the directories in a path name does not exist.   |
| 3 | ESRCH   | No such process<br>The process whose number was given to <i>signal</i> does not exist, or is already dead.  |
| 4 | EINTR   | Interrupted system call<br>An asynchronous signal (such as interrupt or quit), which the user has elected to catch, occurred during a system call. If execution is resumed after processing the signal, it will appear as if the interrupted system call returned this error condition. |
| 5 | EIO     | I/O error<br>Some physical I/O error occurred during a <i>read</i> or <i>write</i> . This error may in some cases occur on a call following the one to which it actually applies.   |
| 6 | ENXIO   | No such device or address<br>I/O on a special file refers to a subdevice which does not exist, or beyond the limits of the device. It may also occur when, for example, a tape drive is not dialled in or no disk pack is loaded on a drive.  |
| 7 | E2BIG   | Arg list too long<br>An argument list longer than 512 bytes (counting the null at the end of each argument) is presented to <i>exec</i> .   |
| 8 | ENOEXEC | Exec format error<br>A request is made to execute a file which, although it has the appropriate permissions, does not start with one of the magic numbers 407 or 410.   |

- 9     EBADF           Bad file number  
Either a file descriptor refers to no open file, or a read (resp. write) request is made to a file which is open only for writing (resp. reading).
- 10    ECHILD          No children  
*Wait* and the process has no living or unwaited-for children.
- 11    EAGAIN          No more processes  
In a *fork*, the system's process table is full and no more processes can for the moment be created.
- 12    ENOMEM          Not enough core  
During an *exec* or *break*, a program asks for more core than the system is able to supply. This is not a temporary condition; the maximum core size is a system parameter. The error may also occur if the arrangement of text, data, and stack segments is such as to require more than the existing 8 segmentation registers.
- 13    EACCES          Permission denied  
An attempt was made to access a file in a way forbidden by the protection system.
- 14    –               (unused)
- 15    ENOTBLK         Block device required  
A plain file was mentioned where a block device was required, e.g. in *mount*.
- 16    EBUSY           Mount device busy  
An attempt to mount a device that was already mounted or an attempt was made to dismount a device on which there is an open file or some process's current directory.
- 17    EEXIST          File exists  
An existing file was mentioned in an inappropriate context, e.g. *link*.
- 18    EXDEV           Cross-device link  
A link to a file on another device was attempted.
- 19    ENODEV          No such device  
An attempt was made to apply an inappropriate system call to a device; e.g. read a write-only device.
- 20    ENOTDIR         Not a directory  
A non-directory was specified where a directory is required, for example in a path name or as an argument to *chdir*.
- 21    EISDIR          Is a directory  
An attempt to write on a directory.
- 22    EINVAL          Invalid argument  
Some invalid argument: currently, dismounting a non-mounted device, mentioning an unknown signal in *signal*, and giving an unknown request in *stty* to the TIU special file.
- 23    ENFILE          File table overflow  
The system's table of open files is full, and temporarily no more *opens* can be accepted.
- 24    EMFILE          Too many open files  
Only 15 files can be open per process.
- 25    ENOTTY          Not a typewriter  
The file mentioned in *stty* or *gtty* is not a typewriter or one of the other devices to which these

calls apply.

- 26    ETXTBSY        Text file busy  
An attempt to execute a pure-procedure program which is currently open for writing (or reading!). Also an attempt to open for writing a pure-procedure program that is being executed.
  
- 27    EFBIG            File too large  
An attempt to make a file larger than the maximum of 32768 blocks.
  
- 28    ENOSPC          No space left on device  
During a *write* to an ordinary file, there is no free space left on the device.
  
- 29    ESPIPE          Seek on pipe  
A *seek* was issued to a pipe. This error should also be issued for other non-seekable devices.
  
- 30    EROFS            Read-only file system  
An attempt to modify a file or directory was made on a device mounted read-only.
  
- 31    EMLINK          Too many links  
An attempt to make more than 127 links to a file.
  
- 32    EPIPE            Write on broken pipe  
A write on a pipe for which there is no process to read the data. This condition normally generates a signal; the error is returned if the signal is ignored.