**NAME**

pipe − create an interprocess channel

**SYNOPSIS**

(pipe = 42.)
**sys pipe**
(read file descriptor in r0)
(write file descriptor in r1)

**pipe(fildes)**
**int fildes[2];**

**DESCRIPTION**

The *pipe* system call creates an I/O mechanism called a pipe. The file descriptors returned can be used in read and write operations. When the pipe is written using the descriptor returned in r1 (resp. fildes[1]), up to 4096 bytes of data are buffered before the writing process is suspended. A read using the descriptor returned in r0 (resp. fildes[0]) will pick up the data.

It is assumed that after the pipe has been set up, two (or more) cooperating processes (created by subsequent *fork* calls) will pass data through the pipe with *read* and *write* calls.

The Shell has a syntax to set up a linear array of processes connected by pipes.

Read calls on an empty pipe (no buffered data) with only one end (all write file descriptors closed) return an end-of-file. Write calls under similar conditions generate a fatal signal (*signal*(II)); if the signal is ignored, an error is returned on the write.

**SEE ALSO**

sh(I), read(II), write(II), fork(II)

**DIAGNOSTICS**

The error bit (c-bit) is set if too many files are already open. From C, a −1 returned value indicates an error. A signal is generated if a write on a pipe with only one end is attempted.