SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD APPLETON LABORATORY

COMPUTING DIVISION

DISTRIBUTED INTERACTIVE COMPUTING NOTE 757

PERQ UNIX IMPLEMENTATION NOTE # 52
Accent/Unix: Pipes

Issued by
J.C.Malone

26 January 1983

DISTRIBUTION:
R W Witty
K Robinson
C Prosser
A S Williams
L O Ford
T Watson
E V C Fielding
J C Malone
P J Smith
A J Kinroy
J M Loveluck
C P Wadsworth
Martin Ritchie (ICL Dalkeith)
P Palmer (ICL Dalkeith)
RL Support/PERQ/Unix Implementation Notes file

A pipe is a method of passing data between two or more agents, usually in different processes.

The pipe system call [2] produces two file descriptors, one associated with the writable end of the pipe, the other with the readable end. Normally a process will then fork another process, which will then share these file descriptors. If, say, the parent process closes his read file descriptor, and the child closes the write end, then the pipe becomes a one-way channel, the parent being able to write data to the pipe, which the child may read.

Note that further forks and dups may be done on the pipe, so that it is possible to have any number of readers and writers on the same pipe. Thus it is necessary for a single, independent process to control the pipe - and this process is, arbitrarily, SwitchBoard.

The system calls [2] to read and write on a pipe are exactly the same as those for files. The only difference is the way the data is handled in SwitchBoard [1]. The written blocks of data are queued and then a read will receive any number of blocks, or the beginning of a block to satisfy its request.

If a read is requested on the read end of a non-empty pipe, it will be satisfied by the number of bytes requested, or the number currently

in the pipe, if this is less. If, however, the pipe is currently empty then the number of writers on the pipe must be checked. No more writers means that there will never be anything written to the pipe, since the end of a pipe cannot be re-opened once the last user has closed it, so the read receives EOF. Otherwise the read request will be queued and the reading process must wait. It will be dequeued either because a writer has provided some data (to satisfy the request), or because the write end of the pipe has now closed down (in which case EOF is returned), or because the requesting process has sent a message to cancel the read.

A write on the write end of a pipe will add data to the end of a list of write buffers, unless there are no more reading processes. A write to a one-ended pipe would never be read, so the writing process receives a SIGPIPE signal.

REFERENCES

1. J.C. MALONE, "Perq Unix Implementation Note # 41 - Accent/Unix: Switchboard", DIC Note # 744, Rutherford Appleton Laboratory (January 83).

2. A.S. WILLIAMS, "Perq Unix Implementation Note # 31 - UNIX System Call Specification", DCS Note # 727, Rutherford Appleton Laboratory (November 1982). [In Preparation].