NAME

popen, pclose – initiate pipe to/from a process

SYNOPSIS

#include <stdio.h>

FILE *popen(const char **command*, **const char** **type*);

int pclose (FILE **stream*);

MT-LEVEL

Unsafe

DESCRIPTION

popen( ) creates a pipe between the calling program and the command to be executed. The arguments to **popen( )** are pointers to null-terminated strings. *command* consists of a shell command line. *type* is an I/O mode, either **r** for reading or **w** for writing. The value returned is a stream pointer such that one can write to the standard input of the command, if the I/O mode is **w**, by writing to the file *stream* (see **intro**(3)); and one can read from the standard output of the command, if the I/O mode is **r**, by reading from the file *stream*. Because open files are shared, a type **r** command may be used as an input filter and a type **w** as an output filter.

The environment of the executed command will be as if a child process were created within the **popen( )** call using **fork**(2), and the child invoked the shell using the call:

Solaris **execl("/usr/bin/sh", "sh", "-c",** *command*, **(char *)0);**
XPG4 **execl("/usr/bin/ksh", "ksh", "-c",** *command*, **(char *)0);**

A stream opened by **popen( )** should be closed by **pclose( ),** which closes the pipe, and waits for the associated process to terminate and returns the termination status of the process running the command language interpreter. This is the value returned by **waitpid**(2). See **wstat**(5) for more details on termination status.

RETURN VALUES

popen( ) returns a null pointer if files or processes cannot be created.

pclose( ) returns the termination status of the command. **pclose( )** returns **-1** if *stream* is not associated with a **popen( )** command and sets **errno** to indicate the error.

EXAMPLES

The following is an example of a typical call:

```
#include <stdio.h>
#include <stdlib.h>
main()
{
        char *cmd = "/usr/bin/ls *.c";
        char buf[BUFSIZ];
        FILE *ptr;

        if ((ptr = popen(cmd, "r")) != NULL)
                while (fgets(buf, BUFSIZ, ptr) != NULL)
                        (void) printf("%s", buf);
        return 0;
}
```

This program will print on the standard output (see **stdio**(3S)) all the file names in the current directory that have a **.c** suffix.

SEE ALSO

ksh(1), **pipe**(2), **wait**(2), **waitpid**(2), **fclose**(3S), **fopen**(3S), **stdio**(3S), **system**(3S), **wstat**(5), **xpg4**(5)

NOTES

If the original and **popen( )** processes concurrently read or write a common file, neither should use buffered I/O. Problems with an output filter may be forestalled by careful buffer flushing, for example, with **fflush( )** (see **fclose**(3S)). A security hole exists through the **IFS** and **PATH** environment variables. Full pathnames should be used (or **PATH** reset) and **IFS** should be set to space and tab (**" \t"**).