III. Pascal/Portable C Interface Library

NAME

Conventions - using the Pascal interface library

SYNOPSIS

```
const
    MAXSTR = 64;    { for instance }
type
    filemode = (PEOF, PEOLN, PINVAL, PVALID, PWRITE, PWROTE);
    filerec = record
        recsize : integer;
        mode : filemode;
        filedesc : 0 .. 255
        end;
    filetype = file of ...;
    openmode = (OPREAD, OPWRITE, OPUPDATE);
    seekmode = (SKFRONT, SKHERE, SKBACK, SKREAD, SKWRITE);
    string = packed array [1 .. MAXSTR] of char;
```

FUNCTION

The Pascal interface library is a collection of routines designed to facilitate use of the portable C library with Pascal programs. This is the only way to perform random access I/O, for instance, or to open a file whose name is not determined until after program startup. Unlike the builtin routines described in the preceding section, none of these routines have been made known to the Pascal translator, so each must be declared properly before it can be used. The declarations shown above provide a proper environment for declaring these interface routines.

The pstat function will deliver up the internal information used to administer open Pascal files. recsize is the size in bytes of the associated buffer variable, or zero for a text file (whose buffer size is known to be one byte). mode is one of the six enumerated states of filemode: PEOF for input file at end of file; PEOLN for input file at end of line; PINVAL for input file with buffer not currently filled; PVALID for input file with buffer filled, and not at end of line; PWRITE for output file with no partial line pending; and PWROTE for output file with a partial line having been output. filedesc is the filedescriptor used by the portable C library for input/output, to which has been added 128 if the file is temporary, and hence must be removed when closed.

filetype is used, in the manual pages of this section, to indicate any kind of file declaration, since the receiving routine is indifferent to the actual Pascal type. Pascal, of course, is not so indifferent; problems arise only when the same interface routine is to be used to deal with two different file types. In this case, the only recourse is to put the different usages in separate files and compile them separately, each with appropriate declarations for the interface routine.

Similar problems can occur with the generic type string, unless it is possible to standardize the maximum string size MAXSTR across all usages in a program. Here the tradeoff is between wasting space and leaving comfortable room for the largest desired strings. The portable C interface presumes that filenames will never exceed 64 characters, counting the terminating NUL (written in Pascal as chr(0)) which must be present. Command

line arguments and other strings may be longer, of course, if used for other purposes.

The enumeration seektype is used only by the function pseek: SKFRONT calls for positioning the file buffer relative to the beginning of the file; SKHERE calls for positioning the file buffer relative to the current location; SKBACK calls for positioning the file buffer relative to the end of the file, if possible; SKREAD changes the file mode to open for reading; and SKWRITE changes the file mode to open for writing.

Many other functions in the portable C library are usable without the assist provided by the Pascal interface library, given some skill in writing Pascal declarations that match up with those for C.

SEE ALSO

Mapping(IV), for an overview of how Pascal declarations are translated.

NAME

argc - command line argument count

SYNOPSIS

```
var
    argc: 0 .. maxint;
```

FUNCTION

argc is set, on program startup, to the number of arguments on the command line. This count includes the name by which the program was invoked, and hence is conventionally always nonzero.

SEE ALSO

argv

NAME

argv - command line argument vector

SYNOPSIS

```
var
    argv: array [0 .. maxint] of string;
```

FUNCTION

argv is set, on program startup, to point at the start of a list of pointers to NUL terminated strings. argv[0] is the name by which the program was invoked, argv[1] is the first argument, if present, and so forth.

SEE ALSO

argc

NAME

getarg - get command line argument

SYNOPSIS

```
function getarg(n: integer; var buf: string; size: integer): boolean;
    external;
```

FUNCTION

getarg copies the NUL terminated string corresponding to argument n, if present, on the command line into the variable buf. At most size characters are copied, counting the terminating NUL; it is not considered an error if the argument is too long to be copied completely.

RETURNS

getarg returns true if the argument is present and has been copied, otherwise false.

EXAMPLE

```
if (not getarg(1, fname, MAXSTR)) then
    writeln('Missing file name');
```

SEE ALSO

argc, argv, nargs

NAME

nargs - get number of command line arguments

SYNOPSIS

```
function nargs: integer;
    external;
```

FUNCTION

nargs returns the number of arguments on the command line, and hence available via calls to getarg. Note that argument zero, the name by which the program was invoked, is also counted. nargs is conventionally non-zero.

SEE ALSO

argc, argv, getarg

NAME

pclose - close a Pascal file variable

SYNOPSIS

```
procedure pclose(var fv: filetype);
    external;
```

FUNCTION

pclose closes the file associated with the Pascal file variable fv. It is not considered an error to close a file that has not been opened.

RETURNS

Nothing.

EXAMPLE

```
pclose (input);
if (not popen(input, fname, false, 0)) then
    writeln ('Can't redirect input');
```

SEE ALSO

pcreate, pinit, popen

NAME

pcreate - create Pascal file by name

SYNOPSIS

```
function pcreate(var filevar: filetype;
    var fname: string;
    mode: openmode;
    recsize: integer): boolean;
    external;
```

FUNCTION

pcreate creates a file with name fname and, if successful, associates it with the Pascal file variable filevar. If mode is not OPREAD, which is almost invariably the case for pcreate, the file is opened for writing; otherwise it is opened for reading. recsize is the size in bytes of the binary record to be held in the buffer filevar, or zero for a text file.

If filevar is already opened, it is quietly closed and reopened.

RETURNS

pcreate returns true if the file is successfully created, otherwise false.

EXAMPLE

```
length := 0;
while (not eoln) do
    begin
    length := length + 1;
    read(name[length])
    end;
name[length + 1] := chr(0);
if (not pcreate(fv, name, true, 0)) then
    writeln('Error, cannot create file');
```

SEE ALSO

pclose, pinit, popen

BUGS

It may be difficult to guess the size of an arbitrary record; rewrite is much smarter in this regard.

NAME

pinit - connect Pascal file variable to C file

SYNOPSIS

```
procedure pinit(var filevar: filetype;
    filedesc : integer;
    mode: openmode;
    recsize: integer);
    external;
```

FUNCTION

pinit associates the file descriptor filedesc with the Pascal file variable filevar, so that conventional Pascal input/output may be performed on files already opened by the portable C routines. If mode is not OPREAD, the file is assumed opened for writing; otherwise it is assumed opened for reading. recsize is the size in bytes of the binary record to be held in the buffer filevar, or zero for a text file.

If filevar is already opened, it is quietly closed before being used. No check is made to see if the file descriptor corresponds to an open file, or if writemode is appropriate. This is the only way to open a file with update mode (reading and writing) and connect it to a Pascal file variable; pseek may then be used to position the buffer within the file and to switch between reading and writing.

RETURNS

Nothing.

EXAMPLE

```
const
    STDERR = 2;
var
    errout: text;
begin
    pinit(errout, STDERR, true, 0);
    writeln(errout, 'This message written to STDERR.');
```

SEE ALSO

pclose, pcreate, popen, pseek

BUGS

It may be difficult to guess the size of an arbitrary record; reset and rewrite are much smarter in this regard.

NAME

popen - open Pascal file by name

SYNOPSIS

```
function popen(var filevar: filetype;
    var fname: string;
    mode: openmode;
    recsize: integer): boolean;
    external;
```

FUNCTION

popen opens a file with name fname and, if successful, associates it with the Pascal file variable filevar. If mode is not OPREAD, the file is opened for writing; otherwise it is opened for reading. recsize is the size in bytes of the binary record to be held in the buffer filevar, or zero for a text file.

If filevar is already opened, it is quietly closed and reopened.

RETURNS

popen returns true if the file is successfully opend, otherwise false.

EXAMPLE

```
if (not getarg(1, name, MAXSTR)) then
    writeln('Error, argument required')
else if (not popen(fv, name, true, 0)) then
    writeln('Error, cannot open file');
```

SEE ALSO

pclose, pcreate, pinit

BUGS

It may be difficult to guess the size of an arbitrary record; reset is much smarter in this regard.

NAME

pseek - administer Pascal read/write file

SYNOPSIS

```
procedure pseek(var filevar: filetype;
    offset: integer;
    mode: seekmode);
    external;
```

FUNCTION

pseek permits random access reads and writes with the Pascal file associated with the file variable filevar. It can be used to position the file so that the next read or write occurs at a specified record, rather than the next record in sequence; pseek can also be used to switch the file mode between reading and writing, assuming the file has been opened for update mode and connected to filevar by a pinit call. The modes are:

SKFRONT - the next read or write should access the record specified by the unsigned integer offset; zero is the first record in the file.

SKHERE - the next read or write should access the record specified by the current record position, to which is algebraically added the signed integer offset. Thus an offset of zero does not alter normal sequential processing of records.

SKBACK - the next read or write should access the record specified by the length of the file in records, to which is algebraically added the signed integer offset. This mode is not supported on all operating system interfaces.

SKREAD - the file mode is changed to permit reading. offset is used as for SKHERE.

SKWRITE - the file mode is changed to permit writing. offset is used as for SKHERE.

Note that random access to text files is not supported by all operating system interfaces.

RETURNS

Nothing. The seek may or may not have succeeded, depending upon the capabilities of the device connected to the file.

EXAMPLE

```
pseek(randfile, recno, SKFRONT);
pseek(randfile, 0, SKREAD);
get(randfile);
randfile.total := randfile.total + newstuff;
pseek(randfile, -1, SKWRITE);
put(randfile);
```

SEE ALSO

pinit, pstat

NAME

pstat - get status of Pascal file

SYNOPSIS

```
function pstat(var filevar: filetype;
    var info: filerec): boolean;
    external;
```

FUNCTION

pstat checks whether filevar is open and, if so, copies its internal status information to the record info.

info.recsize is the size in bytes of the associated buffer variable, or zero for a text file (whose buffer size is known to be one byte). info.mode is one of the six enumerated states of filemode: PEOF for input file at end of file; PEOLN for input file at end of line; PINVAL for input file with buffer not currently filled; PVALID for input file with buffer filled, and not at end of line; PWRITE for output file with no partial line pending; and PWROTE for output file with a partial line having been output. info.filedesc is the filedescriptor used by the portable C library for input/output, to which has been added 128 if the file is temporary, and hence must be removed when closed.

RETURNS

pstat returns true if the file is open, in which case its status is written into info; otherwise it returns false.

EXAMPLE

```
if (pstat(report, inforec)) then
    writeln(report, 'Total is ', total);
```

SEE ALSO

pinit, pseek