CS 240 Programming in C

Input and Output

Oct 6, 2022

Input in C

- Input means to feed some data into a program.
- Input functions: getchar, fgets, scanf
- Output means to display some data on screen, printer, or in any file.
- Input functions: putchar, puts, printf

Buffer vs Stream

- A buffer has a specified, definite length whereas a stream does not.
- A stream is a sequence of bytes that is read and/or written to, while a buffer is a sequence of bytes that is stored.

File streams

FILE *stdin
stdin is associated with a user's standard input stream.
FILE *stdout
stdin is associated with an output stream used for normal program output.
FILE *stderr
stdin is associated with an output stream used for error messages.

Pipe operator '|'

- Pipe character | is used to send the STDOUT from one application to the STDIN of the other application. In Linux the each application that is executed is run in parallel, so each application is processing its STDIN and sending its STDOUT as soon as it is received.
- Usage: command-0 | command-1 | ... | command-n

fscanf, scanf, sscanf

- fscanf - reads formatted input from file.
- scanf - reads formatted input from stdin.
- sscanf - reads data from buffer into the locations given by argument list.

fscanf, scanf, sscanf

```
#include <stdio.h>
int fscanf (FILE *:stream, const char *format, [pointer]);
int scanf(const char *format, [pointer]);
int sscanf(const char *buffer, const char *format, [pointer]);
```

File handling

- Creation of a new file (fopen with attributes as “a” or “a+” or “w” or “w+”)
- Opening an existing file (fopen)
- Reading from file (fscanf or fgets)
- Writing to a file (fprintf or fputs)
- Moving to a specific location in a file (fseek, rewind)
- Closing a file (fclose)

File Pointer

```
FILE *fp = FILE *fopen("demo.txt", "r");
```

fgetc()

```
int fgetc(FILE *stream)
```

fgetc returns the next character of stream as an unsigned char (converted to an int), or EOF if end of file or error occurs.

fgets()

```
char *fgets(char *s, int n, FILE *stream)
```

fgets reads at most the next n-1 characters into the array s , stopping if a newline is encountered; the newline is included in the array, which is terminated by ” . fgets returns s , or NULL if end of file or error occurs.

File Positioning Functions

```
int fseek(FILE *stream, long offset, int origin)
```

fseek sets the file position for stream; a subsequent read or write will access data beginning at the new position.
fseek returns non-zero on error.

File Positioning Functions

```
long ftell(FILE *stream)
```

ftell returns the current file position for stream , or -1 on error.

File Positioning Functions

```
int fgetpos(FILE *stream, fpos_t *ptr)
```

fgetpos records the current position in stream in *ptr , for subsequent use by fsetpos. The type fpos_t is suitable for recording such values. fgetpos returns non-zero on error.

File Positioning Functions

```
int fgetpos(FILE *stream, fpos_t *ptr)
```

fsetpos positions stream at the position recorded by fgetpos in *ptr .
fsetpos returns non-zero on error.

File Buffer

```
int setvbuf(FILE stream, char buffer, int mode, size_t size);
```

Setvbuf changes the buffering mode of the given file stream stream as indicated by the argument mode [1]

[1] https://en.cppreference.com/w/c/io/setvbuf

Error Functions of File Pointers

- void clearerr(FILE *stream):
  clearerr clears the end of file and error indicators for stream .
- int feof(FILE *stream): feof returns non-zero if the end of file indicator for stream is set.
- int ferror(FILE *stream): ferror returns non-zero if the error indicator for stream is set.
- void perror(const char *s) perror(s) prints s and an implementation-defined error message corresponding to the integer in errno.

Macro

A macro is a piece of code in a program that is replaced by the value of the macro. Macro is defined by define directive. Whenever a macro name is encountered by the compiler, it replaces the name with the definition of the macro. Macro definitions need not be terminated by a semi-colon(;). [2]

[2] https://www.geeksforgeeks.org/macros-and-its-types-in-c-cpp/

Macro

A macro is a piece of code in a program that is replaced by the value of the macro. Macro is defined by define directive. Whenever a macro name is encountered by the compiler, it replaces the name with the definition of the macro. Macro definitions need not be terminated by a semi-colon(;). [3]

[3] https://www.geeksforgeeks.org/macros-and-its-types-in-c-cpp/