【Linux Programming】Day20

Tags	
Date	@June 12, 2022
Summary	Reading from and Writing to the Terminal

【Ch5】Terminals

5.1 Reading from and Writing to the Terminal

Canonical vs. Non-Canonical Modes

By default, terminal input is not made available until the user presses Enter or Return. This is usually good because it allows the user to correct typing mistakes using Delete.

This behavior is called canonical, or standard mode.

Until a line of input is complete(usually when the user presses Enter), the terminal interface manages all the key presses, including Backspace, and no characters may be read by the application.

The opposite is called non-canonical mode, where the application has much greater control over the processing of input characters.

Linux uses a line feed alone to end lines of text.

To exclude the carriage return, add the following code:

```
do {
  choice = getchar();
} while (choice == '\n');
```

Handling Redirected Output

It’s common for Linux programs to have inputs and outputs redirected.

However, there are cases where we want to prevent this from happening, or where we want to separate prompts that we want the user to see from other output that can be redirected safely.

We can tell whether the standard output has been redirected by finding out if the low-level file descriptor is associated with a terminal.

```
#include <unistd.h>

int isatty(int fd);
```

The `isatty` system call returns 1 if the file descriptor `fd` is connected to a terminal and 0 otherwise.

Use the following code to make sure that the file’s output isn’t redirected

```
if (!isatty(fileno(stdout))) {
  fprintf(stderr, "The output is not writing to the terminal.\n"P;
  exit(1);
}
```

It’s possible to redirect both `stdout` and `stderr` away from the terminal:

```
$ ./menu >file 2>file.error
```

Or combine the two output streams into a single file:

```
$ ./menu >file 2>&1
```