CS 361: Computer Systems **Lab Session 5 Worksheet**
Fall 2019

Redirecting I/O

1. We will use the **dup2** system call to redirect input and output. You can read about it in section 10.9 of your book, as well as its man page.
2. This system call depends on file descriptors, integers which are returned after open is called on a file. You can learn about them in chapter 10 of your textbook, or review lecture 11 and 12. The important thing to know is that a file descriptor is essentially a shortcut the operating system uses to find a file after you have opened it.
3. Check out the sample code for this lab session at `https://github.com/uicsystems/LabSession5-PracticeCode.git`. Look at the contents of sampletext1.txt, sampletext2.txt, and ioredirect.c
4. Look at ioredirect.c carefully. What do you think will print out for each of the following print statements?
   (a) printf("%s", buf1);
   (b) printf("%s", buf2);
   (c) printf("%s", buf3);
   (d) printf("%s", buf4);
5. Compile and run ioredirect.c. What does the program print? Does it print exactly the way you expect? Make sure you understand exactly what's happening.
6. It is helpful to know that standard in, standard out, and standard error are all automatically given file descriptors when a process starts. Standard in has file descriptor 0, standard out has file descriptor 1, and standard error has file descriptor 2.
7. Answer the questions about dup2 on Gradescope.

Opening files for redirection

1. Your shell will need to support the following file I/O redirection commands:
   (a) **`command > filename`** Redirects the output of command to filename. The existing contents of filename are overwritten.
   (b) **`command >> filename`** Redirects the output of command to filename. The output from command is appended to contents of filename. Existing contents are not overwritten.
   (c) **`command < filename`** Command reads its input from filename instead of from stdin.
2. You should add code to your command parsing loop to detect the redirection commands, and to save the filename. You may assume the filename is under 100 characters.

3. If a file you are writing to does not exist, you should create the file, and set its permissions so it is writable by the user, and readable by everyone.

4. Read the man page for open by typing "man 2 open" and the command line. The man page and our lecture on file I/O should have everything you need to figure out the correct flags and mode to open your files.

Signal Handling

1. The last thing you will need to do for your shell is write signal handlers to catch SIGINT and SIGTSTP.

2. Look at the code in sighandle.c, included in the sample code for this session. What do you think will happen when you run it?

3. Compile and run sighandle.c. Remember SIGINT is generated with Ctrl-C.

4. Answer the questions about signal handling on Gradescope.

5. Add signal handlers for SIGINT and SIGTSTP (generated by Ctrl-Z) to your shell. They should print out "SIGINT Handled." and "SIGTSTP Handled.", respectively.