Due March 6, 2018	**11:59 pm**	**Milestone 1**	
Due March 15, 2018	**11:59 pm**	**Milestone 2**	
Due March 22, 2018	**1:00 pm (DEMO)**	**Milestone 3**	65 points

Goal:
Learn about Unix processes, interprocess communication, fork(), exec(), pipe() dup2(), waitpid().

Description:
You will write a Unix shell, a small command line interface to Unix. The shell will need the ability to launch applications (spawn other processes), do input and output redirection from/to a file < > , allow an application's output to be piped to another application's input **|**, and allow an application to be launched in the background &. You must implement the semicolon to separate commands.

The only *builtin* functions you need to create are **exit** and **cd**.

Simple examples follow. Note the prompt displays the process ID of the shell.

Sample Output:

```
[chadd@arch-small CS460_Shell]$ ./CS460_Shell
21364> ls
CS460_3H.pdf  CS460_3.pdf  hint.txt
21364> cat hint.txt
you should look at man -s 2 pipe
recursion is fun!
21364> cat hint.txt | grep fun
recursion is fun!
21364> cat hint.txt > newFile.txt
21364> ls newFile.txt
newFile.txt
21364> cat < hint.txt | grep man
you should look at man -s 2 pipe
21364> firefox &
21364> ps | grep firefox
 2515 pts/1    00:00:01 firefox
21364> ls ; echo hi
CS460_3H.pdf  CS460_3.pdf  hint.txt
hi
21364> exit
[chadd@arch-small CS460_Shell]$
```

Constraints:

Each command line will be no more than 2047 typed characters. Each symbol (< > |) will be surrounded on either side by at least one space (**cmd1<in|grep** is not allowed). The & will be preceded by at least one space. You do NOT need to support wild cards (**ls *.txt**). The &, if it is present, must be the last item in a command line. A semicolon cannot follow an &.

Functions (and such) you will (probably) need:
fork(), exec??(), strtok_r(), strsep(), dup2(), pipe(), waitpid(), STDOUT_FILENO, STDIN_FILENO

Subversion/GitLab/GitHub (or how do I submit my work?):
You must store your source code in a Subversion repository on zeus or in GitLab/GitHub.
Subversion: Add your project to /home/***login***/SVNREPOS_CS460S18/
Git[Lab|Hub]: New repository CS460S18_Shell

Name your project **CS460_Shell_PUNetID**. Email the repository/share the git repos with me.

Make Targets

CS460_Shell: build the executable named **CS460_Shell**
valgrind: start the executable with valgrind

```
 valgrind  -v --leak-check=yes --track-origins=yes --leak-check=full --show-
leak-kinds=all  ./CS460_Shell
```

You should not see any memory leaks or errors, even across fork()s.

debug: start the executable with the -d command line option
valgrind_debug: start the executable with the -d command line option with valgrind
clean clean

Executable Location

You should build the executable (CS460_Shell) at the root of your Project.

Milestones:

(10 pts) **1:** March 6: Parsing the command line*, handle exit builtin, implement ;

(10 pts) **2:** March 15: Launch an application (with arguments, with backgrounding, no redirection, no pipes), handle cd builtin, implement ;

(45 pts) **3:** March 22: Final Milestone!

*** See the example on the last page.** For the first milestone, you need to be able to display the parse of the command line as shown. Only display this parse if the user starts your shell with the **-d** command line option. This command line option needs to be present in every version you submit. When this command line option is present (even in later milestones) you should parse and display the command *only* and not launch any applications. Use tabs to nest your output. Your terminal tab size may vary from this document.

Notes:

Lookup the parameters that exec??() takes to guide you in building a data structure to represent the parsed command line.

Make sure the use can type either ls or /usr/bin/ls and get the same result.

When in doubt, do whatever /bin/bash does!

This took me about 500 non-comment lines of code.

`See the class schedule web page for helpful GDB links!`

```
[chadd@arch-small CS460_Shell]$ ./CS460_Shell -d

21364> ls
command: ls
     arguments: none
     redirection:
          stdin: none
          stdout: none
     pipe: none
background: no

21364> cat hint.txt > newFile.txt
command: cat
     arguments: hint.txt
     redirection:
          stdin: none
          stdout: newFile.txt
     pipe: none
background: no

21364> cat hint.txt newFile.txt | grep fun &
command: cat
     arguments: hint.txt newFile.txt
     redirection:
          stdin: none
          stdout: PIPE
     pipe: YES
command: grep
     arguments: fun
     redirection:
          stdin: PIPE
          stdout: none
     pipe: none
background: YES

21364>  ls ; echo hi
command: ls
     arguments: none
     redirection:
          stdin: none
          stdout: none
     pipe: none
background: no
command: echo
     arguments: hi
     redirection:
          stdin: none
          stdout: none
     pipe: none
background: no

21364> exit

[chadd@arch-small CS460_Shell]$
```