

# Assignment 3: Simple Shell

---

Exercise C and POSIX programming by writing a simple shell program that provides basic shell commands.

- You cannot use C++. The primary purpose of this assignment is to practice memory management with raw pointer accesses without C++'s wrapper functions.
- Grading will be done with additional inputs that aren't mentioned below. Make sure your program handles corner cases and errors gracefully. Using Google's AddressSanitizer during debugging is recommended.

## Due

---

Submit your implementation before May 14th, Friday, 11:59:59pm, to LMS. We DO NOT allow any late submission.

## Submission

---

Submit a zip file that has your **source code** and **Makefile**. Before the submission, name the zip file with this format: hw3\_STUDENTID.zip (e.g., hw3\_200012345.zip)

# Program Description

---

## 1. Command execution

```
arter97@arter97-x1:~/simplesh$ gcc -o simplesh simplesh.c
arter97@arter97-x1:~/simplesh$ ./simplesh
simplesh:/home/arter97/simplesh$ gcc
gcc: fatal error: no input files
compilation terminated.
simplesh:/home/arter97/simplesh$ ls -ald /tmp
drwxrwxrwt 23 root root 620 Apr 24 21:48 /tmp
simplesh:/home/arter97/simplesh$ /usr/bin/ls -ald /tmp
drwxrwxrwt 23 root root 620 Apr 24 21:48 /tmp
simplesh:/home/arter97/simplesh$ asdf
Failed to exec: No such file or directory
simplesh:/home/arter97/simplesh$ /asdf
Failed to exec: No such file or directory
```

- Your program must be able to execute commands with multiple arguments.
- `execvp()` expects string array as an argument. Your program must be able to split a single string to a string array with whitespace(" \t") as the delimiter. (See man 3 strtok)
- During grading, input length will not exceed 4095 bytes and after splitting, the number of items in the input array will not exceed 63.

## 2. Built-in commands

- Many utilities in Linux are provided as an executable binary form(e.g., `/usr/bin/ls`, `/usr/bin/echo`) and the shell(e.g., `bash`) uses them by calling `exec()`.
- However, there are some features from the shell that are provided as built-in commands such as `cd` and `exit`.

```
arter97@arter97-x1:~/simplesh$ gcc -o simplesh simplesh.c
arter97@arter97-x1:~/simplesh$ ./simplesh
simplesh:/home/arter97/simplesh$ pwd
/home/arter97/simplesh
simplesh:/home/arter97/simplesh$ pwd test
simplesh: pwd: too many arguments
simplesh:/home/arter97/simplesh$ cd /root
simplesh: cd: /root: Permission denied
simplesh:/home/arter97/simplesh$ cd /asdf
simplesh: cd: /asdf: No such file or directory
simplesh:/home/arter97/simplesh$ cd /tmp
simplesh:/tmp$ cd
simplesh: cd: no arguments
simplesh:/tmp$ cd /tmp /usr
simplesh: cd: too many arguments
simplesh:/tmp$ cd ..
simplesh:/$ pwd
/
```

```

simplesh:/$ exit 1 2 3
simplesh: exit: too many arguments
simplesh:/$ exit
arter97@arter97-x1:~/simplesh$ echo $?
0
arter97@arter97-x1:~/simplesh$ ./simplesh
simplesh:/home/arter97/simplesh$ exit 1
arter97@arter97-x1:~/simplesh$ echo $?
1

```

- Command's spec:
  - pwd: Displays currently working directory (cwd)
    - Typical Linux distro includes a non-built-in pwd as an executable binary in /usr/bin/pwd. **You must implement pwd as a built-in command and not execute a separate binary for pwd.**
  - cd: Changes currently working directory
    - Typical Linux's shell changes '~' to the user's home directory (e.g., "cd ~/Downloads" == "cd /home/user/Downloads"). However, you don't have to implement '~' handling in this assignment.
  - exit: Terminates the shell process and optionally returns the provided argument as the return code to the parent process.
    - Bash can show the child process' return code by 'echo \$?'.
- Only pwd, cd and exit will be evaluated for the built-in commands.

### 3. Redirection

- Bash supports redirecting stdin, stdout and stderr to a separate file other than the ones specified from the parent process (typically the terminal screen).

```

arter97@arter97-x1:~/simplesh$ gcc -o simplesh simplesh.c
arter97@arter97-x1:~/simplesh$ ./simplesh
simplesh:/home/arter97/simplesh$ ls -ald /tmp > out
simplesh:/home/arter97/simplesh$ cat out
drwxrwxrwt 23 root root 620 Apr 24 22:42 /tmp
simplesh:/home/arter97/simplesh$ ls -ald /tmp >> out
simplesh:/home/arter97/simplesh$ cat out
drwxrwxrwt 23 root root 620 Apr 24 22:42 /tmp
drwxrwxrwt 23 root root 620 Apr 24 22:42 /tmp
simplesh:/home/arter97/simplesh$ ls -ald /usr > out
simplesh:/home/arter97/simplesh$ cat out
drwxr-xr-x 17 root root 3488 Dec 16 18:43 /usr
simplesh:/home/arter97/simplesh$ ls -ald /usr > /root/out
Failed to open file for stdout redirection: Permission denied
simplesh:/home/arter97/simplesh$ ls -ald /usr > /asdf/out
Failed to open file for stdout redirection: No such file or directory
simplesh:/home/arter97/simplesh$ exit

```

- In this assignment, only stdout redirection is required.
- You need to support both truncate ('>') and append ('>>') mode.
- You do not need to support any other additional syntaxes.
- **Redirection doesn't need to work on built-in commands.**

# Recommended Functions

---

- perror(3): print a system error message
  - exit(3): cause normal process termination
  - strtok(3): extract tokens from strings
  - getcwd(3): get current working directory
  - atoi(3): convert a string to an integer
  - fork(2): create a child process
  - strstr(3): locate a substring
  - open(2): open and possibly create a file
  - dup2(2): duplicate a file descriptor
  - close(2): close a file descriptor
  - execvp(3): execute a file
  - wait(2): wait for process to change state
- It is highly recommended to read the manpages for these functions.
    - e.g., man 3 getcwd

## Note

---

- **You cannot use C++ for this assignment.** The primary purpose of this assignment is to practice memory management with raw pointer accesses without C++'s wrapper functions.
- The included skeleton code will not compile without including headers. Please add the appropriate headers after reading the manpage.

```
arter97@arter97-x1:~$ man -k strcmp      # Search strcmp in manpages
FcStrCmp (3)          - compare UTF-8 strings
FcStrCmpIgnoreCase (3) - compare UTF-8 strings ignoring case
strcmp (3)            - compare two strings
arter97@arter97-x1:~$ man 3 strcmp      # See the strcmp manpage
```

- **Using the system() function from stdlib.h is prohibited.**
- Leaked memory or file-descriptors will be taken into account when grading. Please use Google's AddressSanitizer to prevent the former and the included check\_fd() function in the skeleton code to prevent the latter.
- The provided skeleton code and recommended functions are just examples. **You can diverge from the skeleton code** as long as it matches the description above.
- All inputs must be sanitized (e.g., not entering any arguments on a command that requires one or entering multiple arguments when a command requires only one) and **handle errors** (e.g., cd'ing to a non-existent directory or redirecting to a file with no write permission) gracefully.
- **Special characters** that would normally require escaping (e.g., mkdir dir\ with\ space, cd "dir with space") **won't be used during grading.**