Principles of Operating Systems

CMSC 421 - Spring 2018

Homework 2

50 points total

Due by 11:59PM EDT on Saturday, April 28, 2018

Part I

25 points

Please provide written responses to the following questions. Each question is worth 5 points.

1. Describe the functions of the Memory Management Unit in a modern computer system.
2. What is a virtual address space? Why do we use virtual address spaces in modern operating systems?
3. Compare and contrast the use of paging and segmentation for memory management in an OS.
4. Consider a logical address space of 64 pages of 1024 words each, mapped onto a physical memory of 32 frames of 1024 words each. How many bits are required to represent a logical address (assuming word-based addressing).
5. With the same address space layout as the previous question, how many bits are required to represent a physical address (once again, assuming word-based addressing).

Part II

25 points

Complete the following programming assigment and submit your code using the GitHub repository that you will have created for this assignment. This assignment must be completed in the C programming language (you can choose to use C89/C90/ANSI-C, C99, or C11 as you see fit).

In this assignment, you will be augmenting your shell from Homework 1 with a few new features:

- A built-in `exit` command that can take zero or one arguments that exits your shell. If an argument is specified, it must be an integer, which should be used as the exit code for the program. If a non-integer argument or multiple arguments is given, you must simply print out an error message and not exit. If no argument is given, assume that the exit code should be 0.
- A built-in `chdir` command that changes the current working directory of the shell. If no argument to this command is specified, then the directory shall be changed to the user's home directory (assuming `HOME` is set in the user's environment — if it is not set, then you shall print out an error). You must also be sure to also update the `PWD` environment variable. You may assume that no paths given will ever exceed the value of `PATH_MAX` which is defined in `<limits.h>`. You may ignore anything after the first fully unescaped argument to the command (so `chdir /long/"space filled"/path/name abc` should just ignore the `abc` part).
- A built-in `cd` command that shall do the same thing as the `chdir` command specified above.
- A built-in `echo` command that shall unescape the given string and print it out, followed by a newline on the user's standard output.
- All built-in commands shall have priority over identically named programs that are in the user's `PATH`. That is to say, if the user has an `echo` program in his or her path (say in `/bin`), unless they specifically call that program by a full path, you should run your own internal implementation and not the one provided by the program. If the user specifies a full path, like `/bin/echo` (in this example), then you'd run the program and not your implementation.
- All built-in commands shall unescape commands where appropriate. That is to say that the argument to `chdir` and `echo` shall be unescaped prior to use. In addition, all program paths shall be unescaped and all arguments to non-

built-in programs should also be unescaped properly.

To aid in your development of this new version of the shell, I've provided some useful utility functions. You can find the code to these functions here and a header file for them here. In particular, the `unescape` function in there should be quite useful (i.e, this will do the hard work of the last requirement up there). You do not have to use these source files if you don't want to, but it would be kinda silly not to. If you choose to not use them, you must still unescape strings as specified above in the same manner as my `unescape` function. It will be up to you to ensure that your implementation is functionally equivalent if you do not use mine.

Your shell program is not allowed to use any external libraries other than the system's C library and the files provided in this assignment. Do not try to use libraries like Readline. You will lose points for using external libraries to implement shell functionality! In addition, the C library functions specifically outlawed in Homework 1 are still outlawed in this assignment, as is using another shell to do the work of parsing arguments and running commands. Please refer back to Homework 1 if you have any questions on this.

Here is an example shell session demonstrating some of the new functionality in this assignment:

```
$ ls -l
total 24
drwxr-xr-x  14 lj  staff   476 Mar  3 16:44 shell
-rwxr-xr-x   1 lj  staff  9420 Feb 18 11:42 simple_shell
$ cd shell
$ /bin/ls -la
total 184
drwxr-xr-x  14 lj  staff    476 Mar  3 16:44 .
drwxr-xr-x   5 lj  staff    170 Feb 19 14:59 ..
drwxr-xr-x  13 lj  staff    442 Feb 19 12:53 .git
-rw-r--r--   1 lj  staff    304 Mar  3 11:02 Makefile
-rw-r--r--   1 lj  staff   5103 Mar  3 16:44 builtin.c
-rw-r--r--   1 lj  staff    228 Mar  3 10:58 builtin.h
-rw-r--r--   1 lj  staff  10808 Mar  3 16:44 builtin.o
-rwxr-xr-x   1 lj  staff  20048 Mar  3 16:44 simple_shell
-rw-r--r--   1 lj  staff   2674 Mar  3 16:27 simple_shell.c
-rw-r--r--   1 lj  staff   8272 Mar  3 16:27 simple_shell.o
-rw-r--r--   1 lj  staff   3568 Feb 23 11:34 test.o
-rw-r--r--   1 lj  staff   7558 Mar  3 16:10 utils.c
-rw-r--r--   1 lj  staff   2330 Mar  3 16:19 utils.h
-rw-r--r--   1 lj  staff   9368 Mar  3 16:10 utils.o
$ chdir /
$ ls home
lj
$ chdir
$ ls
shell        simple_shell
$ echo \x48\151\x20\157\165\164\040\x74\x68\x65\x72\x65\041
Hi out there!
$ echo Goodbye, \'World\'\a
Goodbye, 'World'
$ /bin/echo \x48\151\x20\157\165\164\040\x74\x68\x65\x72\x65\041
Hi out there!
$ exit
```

When submitting your shell program, please be sure to include the source code of the shell program (in one or more C source code files), as well as a Makefile that can be used to build the shell. Your shell must be able to be built and run on a VM as has been set up for this course in your projects. If you use my utility functions, make sure to include those files as well.

As this should be an extension of your earlier homework 1 shell, you should be using the same directory and git repository for it. To submit your code, you should do the following:

```
git add your_updated_and_added_files_go_here
git commit
git push -u origin master
git tag hw2
git push origin --tags
```

Last modified Tuesday, 17-Apr-2018 22:45:22 EDT