CSCI 4404/5401 - Assignment-1
Due: 18th September, 2019 (Wednesday)

Please carefully read and adhere to **all** the below guidelines:

1. All students can score 120 points in this assignment.
2. This Assignment is for **100 points** for **CSCI 4401** students. The other 20 points are bonus for CSCI 4401 students. It is highly recommended that 4401 students work on all the questions in the assignment as this will increase your chances of getting the full 100 points.
3. This Assignment is for **120 points** for **CSCI 5401** students. They are required to attempt all the questions.
4. The Assignment scores will be reduced to 10 points for the final grade. For example, if a CSCI 4401 students scores 95 points in this assignment, the student will get 9.5 points credit towards the course grade. But, if a CSCI 5401 student scores the same 95 points in this assignment, the student will get (95/120) * 10 = 7.9 points credit.
5. You will need a Linux OS to do the questions in the assignment. Use a VM application (for example: VirtualBox) to install one if you have a non-Linux system such as a Windows or a Mac. Personally, I prefer to use Debian-based OSs such as Ubuntu.
6. All answers (such as text or short commands) and images should be added to a **single** **PDF** file with title **YOUR_LAST_NAME_FIRST_NAME.pdf**. For example, my file should be titled vadrevu_phani.pdf. This file will be referred to as "**Main PDF**" file through out the rest of this document.
7. Source code should be in **separate files**. (Not in Main PDF file). **Make sure to include the question numbers in the names of all the source code files.** These **files should also be referenced in the Main PDF file**.
8. Put all the files in a directory named **YOUR_LAST_NAME_FIRST_NAME**, compress it (zip or gz) and upload the compressed file to Moodle.
9. There is a **strict late submission policy** for the assignment: **15% points** will be **deducted** for **every day** that the assignment is late.
10. The due date for the assignment is **18th September 2019 (Wednesday), 11:55 PM**

Question-1 [30 points]:

Skills tested:

- Ability to explore “man” pages for new shell commands
- Making multiple shell commands work together

Your task here is to draft a single line shell command that depicts a colorful animal uttering a Chuck Norris fact.You will need to do this using the following commands:

1. lolcat
2. cowsay
3. jq (https://stedolan.github.io/jq/)
4. curl (to fetch a Chuck Norris joke from a web server)

Rules:

1. You should use **all** the commands given above.
2. You **cannot use any other** commands.
3. Each time your command is run, a Chuck Norris joke should be fetched randomly by using this API - http://www.icndb.com/api/ . You should use **curl** to fetch the joke. There are some crude and vulgar jokes in icndb jokes repository. If you want to avoid these offensive jokes, consider searching for the “nerdy” category jokes and excluding the “explicit” category jokes when using the API.
4. The API returns a JSON file which needs to be parsed with **jq** command.
5. Use **cowsay** to draw the animal. Cowsay has many different animal options to pick. Read cowsay’s man page to figure out what these options are. You can pick **any option that starts with the first letter of one of your names**. For example, in my case I have 2 letter options: P and V as my name is Phani Vadrevu. I found that there are 2 options I can pick: “pony” or “vader” (for Darth Vader, the Star Wars character). Similarly, **everyone needs to pick their own animal to get full credit** for this question. **Don’t submit your answers with the default animal** provided by cowsay.
6. Make sure **no new file is being written** to disk permanently as a result of running this command.
7. Use **lolcat** to color the animal like a rainbow.

8. Except for curl, the other tools are probably not installed on your machine. Please install them before trying to use them. Debian-like OS users can use the apt-get command to install these tools easily.

Submission Requirements:

The **final one-line shell command** and **a screenshot of one of the random jokes** produced should be added to the Assignment's **main PDF file**. For your reference, one of the random Chuck Norris facts my command produced is shown below:

Question-2 [40 points]

Skills tested:

- Understanding and making system calls such as fork(), pid(), ppid() etc.
- General programming knowledge: Getting familiar with a graphing library (such as graphviz)

Please do the below tasks:

Task 1**: Write** C-language code that does `fork()` calls in a loop as below:

```
for (i= 1 to n) {
    fork()
}
```

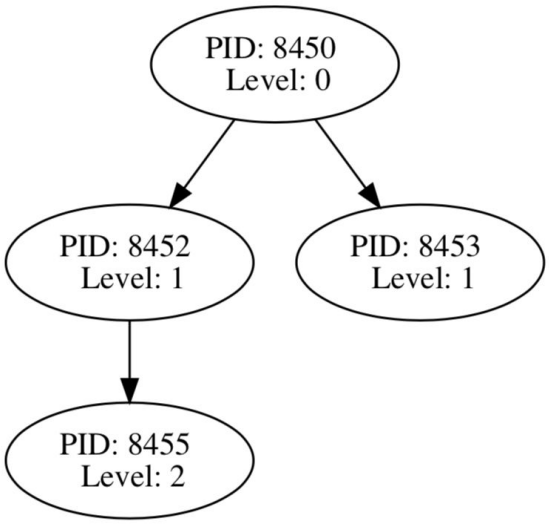
The value of **n should be input** from the user. The program should also **output the following to a file** for each process:

- **Process ID**
- **Parent process ID**

Note: Sometimes, a parent process exits before the child makes a system call to find the parent process ID. In these cases, you will get an incorrect parent process ID. In order to prevent this you can either have your code sleep for a little bit (like a second) before exiting or use a waitpid() call at the end of your code.

Task 2**: Write another** program (in a language of your choice) to parse the file that was output above and **build a Process Tree**. Every node should represent a process and must display its **Process ID**. Every node should also have a "**Level**" field which indicates how far it is (in terms of nodes) from the root node of the process tree. For example, the grand child of the process tree's root will have a "Level" of 2.

Here's an example of a Process Tree that your program might build if the parameter n is 2. I built this using the awesome graphviz library and a Python interface for it. (You can do a little bit of digging and find ways to use graphviz with C, C++ or Java as well). Both the Level and Process ID should be shown in each node.



Submission Requirements:

1. Include Task-1's source code as a separate file in submission.

2. Include Task-2's source code as a separate file in submission.
3. Run Task-1's code with n=4. Then, use the output file of Task-1 to run Task-2. Include the output file from Task-1 and network graph from Task-2 in the Main PDF file.
4. Repeat 3. above for n=5.

Question-3 [10 points]

Skills tested:

- Understanding of fork(), waitpid().

Modify your code from Question-2's Task-1 to make sure that the parent waits for the child to die soon after the fork and then breaks from the loop and exits. Here's some skeletal pseudo code for that loop:

```
for (i= 1 to n) {
    pid = fork()
    .....
    if parent:
        waitpid(pid)
        break
}
```

Submission Requirements:

1. Add this modified source code for Task-1 as a separate file.
2. Run the above modified source code on n=3 and n=5. Then, run Task-2's code (from Question 2) on the output file produced by the modified source code. Include these 2 modified network graphs output by Task-2's code in the Main PDF file.

Question-4 [20 points]:

Skills tested:

- Understanding of fork().

Modify your code from Question-2's Task-1 to make it print "Level" as defined in Question-2's Task-2. This will mean that each process needs to keep track of what level it is in by itself. Make sure that these "Level" values match the "Level" values that would be computed by Question-2's Task-2 code.

Submission Requirements:

1. Add the modified source code for Task-1 as a separate file.
2. Run the modified source code on n=3 and n=5. Include the outputs from these runs in the Main PDF file.

Question-5 [20 points]:

Skills tested:

1. Reading and understanding Chapter-1 material from other sources. Connecting this with what we learnt in class.

Read this article and answer the questions that follow:
https://thorstenball.com/blog/2014/06/13/where-did-fork-go/

1. Why did the article's author not see fork() calls in strace output?
2. Which system call did the author see instead of fork() in strace?
3. What is the library whose library procedure was called by the author's code?
4. The below diagram from the textbook shows how the read() call is performed. Draw a similar diagram describing the fork() call that the author made. **Hint**: This will be simpler than textbook's diagram as fork has no parameters. (So, no step 1, 2 or 3). However, unlike in the textbook, step 4 and step 5 will not have matching names due to the reason you mentioned for Part-1 above. Make sure you show these different names clearly in your answer.

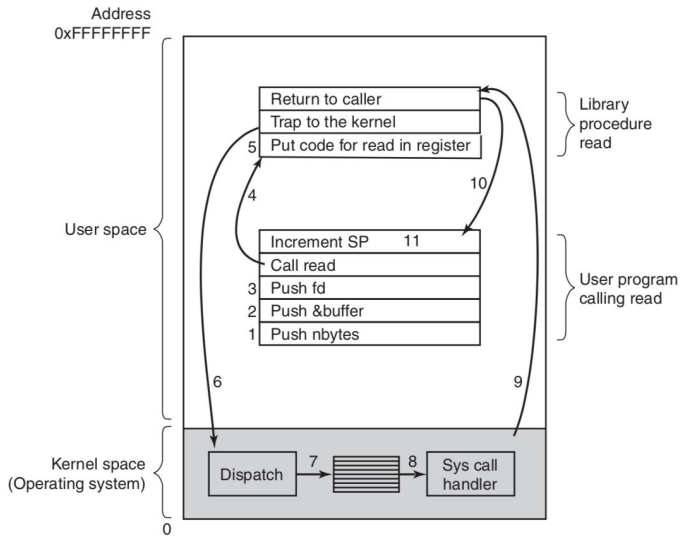


Figure 1-17. The 11 steps in making the system call read(fd, buffer, nbytes).

Submission Requirements:

1. Include all the answers for the 4 questions above in the main PDF file. Question 4 will need a diagram which should also be included in the PDF file.