**NCSE-381: Systems 2**

**Homework #3 Part A**

**Due: Wed Sept. 18 2019 before 11:59 PM (Midnight)**

**Email-based help Cutoff: 5:00 PM on Tue, Sept 17 2019**

Maximum Points for This Part: 18

|  |
| --- |
| **Submission Instructions**  This homework assignment must be turned-in electronically via Canvas. Type in your responses to each question (right after the question) in this MS-Word document. You may use as much space as you need to respond to a given question. Once you have completed the assignment, save it as a PDF file, and upload the PDF to Canvas.    **Note that copy-pasting from electronic resources is plagiarism. Consequently, you must suitably paraphrase the material in your own words when answering the following questions.** |

|  |  |
| --- | --- |
| **Name:** | Noah Dunn |

|  |
| --- |
| **Objective** |
| The objective of this homework is to review and recapitulate the basic concepts related to operating systems and C++ programming. |

Review the Chapter #1 and Chapter #2 from the reference book “Operating System Concepts” (Link available off Syllabus page on Canvas) before proceeding with this exercise.

**Mandatory videos to study** -- **you will get questions in exams based on the content in these videos**:

1. Introduction to OS: <https://youtu.be/PPSmAlnrJxo>
2. Booting: <https://youtu.be/7D4qiFIosWk>
3. Briefly describe (with at least 4 to 5 sentences for each) the two major functions of an operating system using suitable examples for each one of the scenarios (watch first video) [**2 points**]
   1. Function #1 (with 2 examples):

|  |
| --- |
| One major function of operating systems is user management, commonly defined as a way of managing users and their accesses to different items on a system. A user will often be given a unique identifier, normally referred to as a uid. They can then be added to what is called a group, each with their own unique identifiers, commonly called gids. From there, protection and access to different files and directories can be protected according to gid. |

* 1. Function #2 (with 2 examples):

|  |
| --- |
| A second major function to the usage of an OS is User Management. This grouping is what establishes which program need run and what resources these programs can use. Process Management mandates and removes the abstraction of programs into the specific related instructions. This grouping also handles all monitoring related to these programs and the termination of the processes and their resources. |

1. The following questions relate to Boot Loaders (from second video) [**2 points**]
   1. What is a boot loader and what is its primary purpose?

|  |
| --- |
| The boot loader is what starts the first instruction loaded from the very first sector from a data storage device like a Hard Drive or USB. The boot loader itself is just a small program. |

* 1. Where is a boot loader found on a permanent storage device such as a floppy disk?

|  |
| --- |
| The boot loader is in the first sector of a permanent storage device like a Floppy. |

* 1. What is a 2-stage boot loader?

|  |
| --- |
| 2-stage boot loaders first load the actual loader for an operating system(called the stage-2 loader), then they load the Operating System |

* 1. Why is 2-stage boot loading used?

|  |
| --- |
| When an operating system is so big it requires a loader that is larger than 512 bytes. |

1. What is a system call and how does it operate? **[1 point]**

|  |
| --- |
| Any call to the Operating System’s API is a System Call. The operating system will expose this API to the other resources running on a computer, enabling system calls to be executed from these resources. It is used in developing drivers and other software applications. |

1. Describe three general methods for passing parameters to the operating system in a system call (**exactly the same as question 2.13 from textbook so read the relevant section in the textbook**)  **[2 points]**

|  |
| --- |
| 1. Passing parameters as registers 2. Storing parameters in a table or “block” in memory and then passing the address of the block into a register as a parameter.  3. Pushing parameters into a stack by the program, and popping them off the stack using the operating system and passing them as parameters |
|  |

1. From a Linux perspective, briefly (2 sentences) describe the use of user id (a number) and group id (a number) [**1 points**]

|  |
| --- |
| A userid allows us to associate every single user who has access to our system with a unique identifier. A groupid allows us to collect those users into groupings and assign privileges to access certain processes and files based upon group permission. |

1. Table at least 2 significant differences between batch processing and multiprocessing. [**1 points**]

|  |  |
| --- | --- |
| ***Batch Processing*** | ***Multiprocessing*** |
| Processes 1 Job at a time | Multiple Jobs at the same time |
| Jobs own all resources while they run | Processes take turns with the CPU |
|  |  |

1. Briefly discuss a realistic scenario when designing an operating system as a virtual machine would be advantageous. [**1 points**]

|  |
| --- |
| If we had a well-designed system we wanted to replicate multiple instances of, with multiprocessing in tow, and with no system to system interactivity, a virtual machine would be our game plan. |

1. What is virtual memory? Why is it used on most modern operating systems? [**1 points**]

|  |
| --- |
| Virtual Memory is an abstraction of real memory, used by the OS + CPU as way of managing underlying Physical Memory, while not having to have programmer worry about managing addressing and physical locations of all bytes created. |

1. What is the difference between a Type-1 and a Type-2 hypervisor? [**1 points**]

|  |
| --- |
| Type-1 Hypervisor do not require a native OS and run straight from the hardware of a computer. Type-2 Hypervisors run within the context of the native OS. |

|  |
| --- |
| ***Review programming from CSE-278*** |
| The objective of the next few questions is to review pertinent programming concepts from CSE-278  Prior to answering the questions in this homework it may be beneficial to briefly following chapters from the E-book titled “[C++ How to Program](http://proquest.safaribooksonline.com.proxy.lib.miamioh.edu/9780134448930)” (Link to e-textbook is in the Syllabus on Canvas):   |  |  | | --- | --- | |  Chapter 1 (Introduction) |  Chapter 2 (Expressions & I/O) | |  Chapter 3.1, 3.2 (String & methods) |  Chapter 4 (if-statements) | |  Chapter 5.1 – 5.10 (loops & switch) |  Chapter 6 (functions/methods) | |

1. What is path to a file? Using suitable examples explain the differences between relative and absolute path? (answer in 2 or 3 separate sentences) [**1 point**]

|  |
| --- |
| The path to a file is the series of steps we have to take to gain access to that file. A relative path such as Downloads/doc1.docx can be identified by the lack of a preceding “/” before the first folder, and will direct us from where we currently are until the path is finished. An absolute path, normally beginning with a “/” will start from the beginning or “root” of a file system. I.E /home/dunnnm2/public\_html/383/html1.html. |

1. Briefly (2 to 3 sentences each) discuss pass-by-value versus pass-by-reference mechanisms for passing parameters to methods. As per this course's conventions, which approach is preferred based on data type of parameters in C++? [**2 points**]

|  |
| --- |
| Passing a variable by value asserts that whatever changes are made when we throw that value into a method, will only be changed within the confines of the method. Any changes to the value will be unable to be access upon the conclusion of the method. Passing by reference means the exact opposite, any change we make to the variable inside the method will be made outside the method as well. All changes will hold true at the conclusion of the method body. Passing by value is preferred for primitives, while passing by reference is preferred for objects. |

1. List at least 3 unique properties that can be inferred from data type of a variable? [**1 point**]

|  |
| --- |
| 1. Size(in Bytes)  2. Holdable Values(1, true, “blah”)  3. Usable Methods(.substr(), .toString()) |

1. What is the size\_type or size\_t data type? Illustrate its use with an example of a for-loop (do not write whole program, but just 1 line of the for-loop) [**1 point**]

|  |
| --- |
| Size\_T is simply an alias or “renaming” of the datatype unsigned long. It is commonly associated with string lengths.  Example: for(size\_t i = 0; i < str.length(); i++){ |

1. Given a constant string object str (const std::string str) what methods can be called on str? Give an example of 1 method that can and 1 method that cannot be called on the object str? [**1 point**]

|  |
| --- |
| 1. Can call: str.substr() 2. Cannot call: str.resize() |