

## HomeWork 2, OS Principles

**Due Date:** Wednesday 3/10/2021  
**HW Delivery:** submit on **Canvas** by the due date, before midnight  
**Total Points:** 60

**General rules:** Create homework, compose specifications using a common *document-creation* tool, such as Microsoft® Word. Each question is worth 5 points.

**Hints:** Refer to the wwweb or lecture notes for this class to answer the questions below. Be concise, complete, and precise.

1. Characterize high-level the **Solaris OS** design. State some key advantages of its design.

Loadable kernel modules used in Solaris OS, allows for the kernel to provide core services, while other services are implemented dynamically as the kernel is running. This preferred over adding new features directly to the kernel. When compared to other hardware products Solaris OS design is much more portable.

2. Write a concise, **complete** English description of the C/C++ function **system()**. Be thorough.

**System ()** is part of the C/C++ standard library, meaning in order to call the function we need to include <stdlib.h> into our program. It takes in a single string as an argument and this argument can be any Linux/UNIX command. Lastly this function is executed in the terminal of the operating system and will return the status of the executed command and have status of zero if executed without any errors.

3. Write a concise English language description of the C/C++ function **fflush()**.

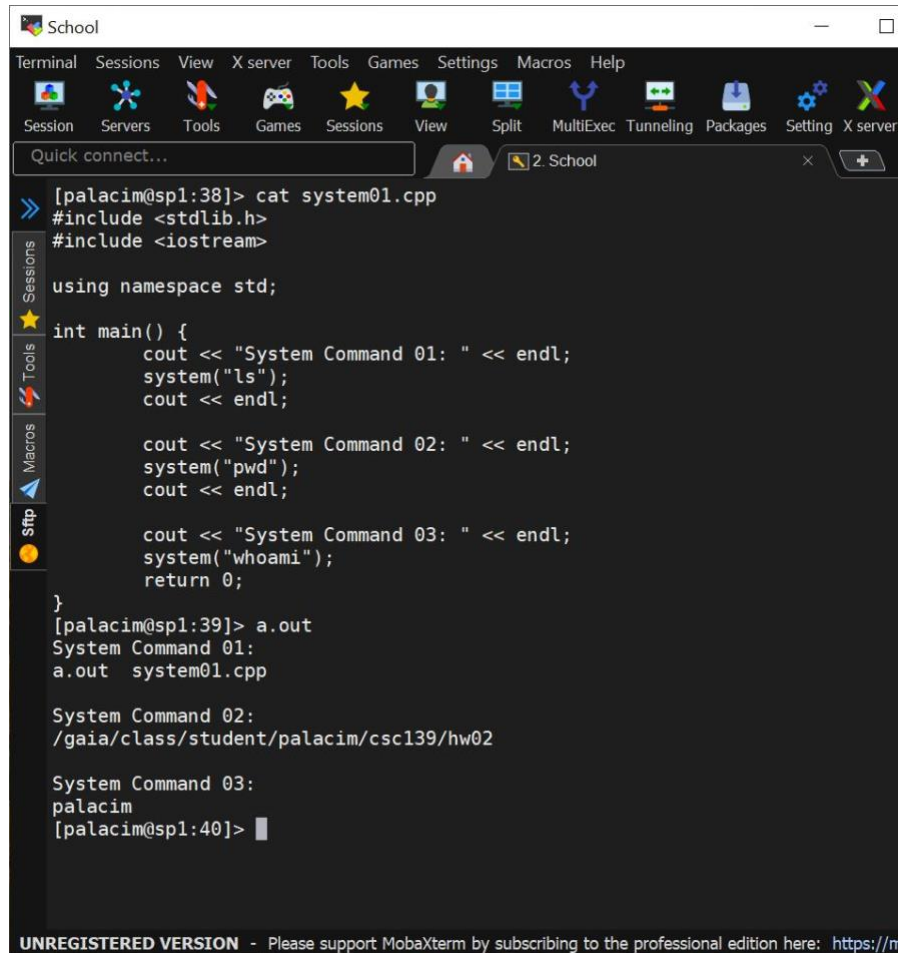
**fflush ()** is part of the standard for C/C++ and must include its library in order to execute the function. Used typically for output streams only. Its' purpose is to clear (flush) all user-space buffered data. Any unwritten data in its' output is written to the file.

4. Write a concise, complete English language description of the OS command **whoami**. Be thorough but write only few lines of printed text. Also try WHOAMI on MS Windows; describe and document what you observe.

**whoami** is found on majority UNIX/UNIX-like operating systems, where if executed prints out username of the current user. On a MS Windows **whoami** prints out more specific user information. For example "desktop=jj\*\*\*\*r\palc" where the \*\*\*\* are numbers specific to my machine.

5. Write C++ program **system1** that calls library function "**system()**". Issue 3 distinct calls to `system()`, each with a single argument: "`ls`", "`pwd`", and "`whoami`". Show the source and all generated outputs. Describe briefly what you observe.

When the program was ran, I saw a list of files in the current directory I was in for "`ls`." Then when "`pwd`" was executed I saw the path for the current directory. Lastly when "`whoami`" was executed it should the username of who I was logged in as.



```
[palacim@sp1:38]> cat system01.cpp
#include <stdlib.h>
#include <iostream>

using namespace std;

int main() {
    cout << "System Command 01: " << endl;
    system("ls");
    cout << endl;

    cout << "System Command 02: " << endl;
    system("pwd");
    cout << endl;

    cout << "System Command 03: " << endl;
    system("whoami");
    return 0;
}
[palacim@sp1:39]> a.out
System Command 01:
a.out system01.cpp

System Command 02:
/gaia/class/student/palacim/csc139/hw02

System Command 03:
palacim
[palacim@sp1:40]>
```

6. Write C or C++ program **system2** that reads OS commands via command line parameters, and then **executes them**. The command line parameters must be legal Unix/Linux commands. Print the number of commands entered. To prepare, read about **argc**, **argv**, and **envp**. Focus is only **argc** and **argv**. A sample execution by fictitious user "`herb`" is:

```
herb$ ./a.out pwd ls whoami
```

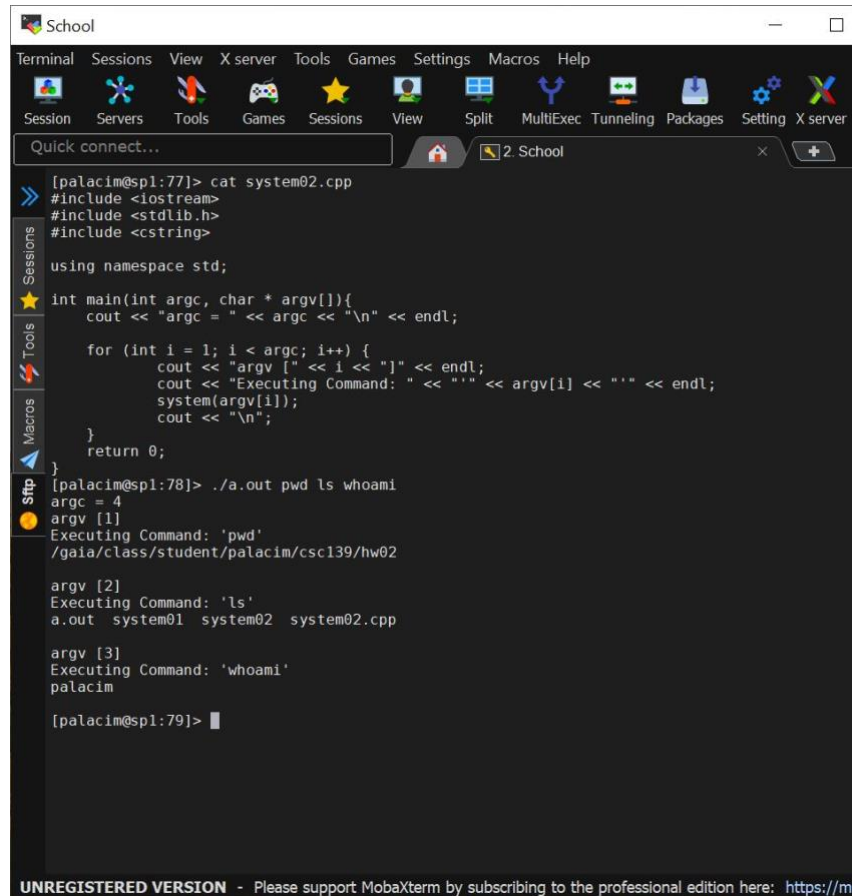
```
argc = 4
```

```
argv[1] = pwd
Executing command 'pwd'
/Users/herbertmayer
```

```
argv[2] = ls
Executing command 'ls'
```

Applications	Downloads	Music	familie	test1
Contacts	Favorites	Pictures	herb	test1.c
Desktop	Library	Public	pix_low_2020	test2
Documents	Movies	a.out	test.c	test2.c

```
argv[3] = whoami
Executing command 'whoami'
herbertmayer
```



```
[palacim@sp1:77]> cat system02.cpp
#include <iostream>
#include <stdlib.h>
#include <cstring>

using namespace std;

int main(int argc, char * argv[]){
    cout << "argc = " << argc << "\n" << endl;

    for (int i = 1; i < argc; i++) {
        cout << "argv [" << i << "]" << endl;
        cout << "Executing Command: " << argv[i] << endl;
        system(argv[i]);
        cout << "\n";
    }
    return 0;
}

[palacim@sp1:78]> ./a.out pwd ls whoami
argc = 4
argv [1]
Executing Command: 'pwd'
/gaia/class/student/palacim/csc139/hw02

argv [2]
Executing Command: 'ls'
a.out system01 system02 system02.cpp

argv [3]
Executing Command: 'whoami'
palacim

[palacim@sp1:79]>
```

7. What is an Operating System (OS)? What is it used for? What are the goals of an OS?

An OS manages resources and information that is entered and received by the user's request. Its goal is to use resources efficiently and fairly, to prevent errors and improper use.

8. What are 5 key functions and responsibilities of an OS? List, name, and describe them.

- 1). User interface – allows for user to enter and receive information. Came be in the form of textual or graphical
- 2). Manage resources – decides between conflicting request for efficient and fair resource use
- 3). Manage tasks -
- 4). Manage files - create, delete, copy, rename, print, list, and generally access and manipulate files and directories
- 5). Provide utilities – programs that help perform individual, specialized management tasks

9. OS commands can be visual or purely textual, or completely graphical. Explain pros and cons. Name a sample OS for both types.

Textual OS commands are mostly used in UNIX, where it makes commands simple to implement and can be captured as files. However, there are a large set of commands that not an average user may want to remember or try to understand.

Graphical OS commands can be found in touchscreen devices, where it can be classified as user friendly and is intuitive anyone can operate. However, they are slower than textual OS commands, difficult and expensive to develop.

10. What is the meaning of **Information Hiding**? How is this related to OS? State your assessment, regarding OS design.

**Information hiding** is when design decisions are hidden so certain code cannot be modified or changed. One advantage to this is allowing a programmer to modify a program. It also makes it easier for a user to learn, by managing complexity. In OS design this is important because it allows source code to be placed in modules, and as the program develops and grows, it will make it easier to access.

11. Briefly contrast ages, origins, uses of: MS-DOS, Unix BSD, Linux

MS-DOS – the use is very simple and includes single memory space. Introduced August 1981 by Microsoft

Unix BSD – originated from Berkley created around the 70s. It is used for multitasking

Linux – one of the main uses is its reliability, zero cost, and free distribution. It is newly developed, by Linus in 1991

12. Name and briefly outline some ideal, high-level OS design goals.

Some high-level OS design goals include:

- being able to run on multiple platforms
- have a low cost but high performance (low overhead)
- have an overall consistent user interface, that is user friendly