TA responsible for HW1: Jingcan Chen (jchenhv@cse.ust.hk)

Fall 2024 COMP 3511 Homework Assignment #1

Handout Date: September 15, 2024, Due Date: October 2, 2024

Name	
Student ID	
ITSC email	@connect.ust.hk

Please read the following instructions carefully before answering the questions:

- You must finish the homework assignment **individually**.
- This homework assignment contains **three** parts: (1) multiple choices, (2) short answer 3) programs with fork()
- **Homework Submission:** Please submit your homework to **Homework #1** on **Canvas**.
- TA responsible for HW1: Jingcan Chen, jchenhv@cse.ust.hk

1. [30 points] Multiple Choices

Write your answers in the boxes below:

MC1	MC2	MC3	MC4	MC5	MC6	MC7	MC8	MC9	MC10

1) Which of the following services is NOT necessary for an operating system?
A) File System
B) Error Detection
C) GUI
D) I/O Operation

2) Which of the following statements about *interrupt* is **not true**?
A) Interrupts are used to handle asynchronous events, e.g., I/O ops and software errors
B) Interrupts can have different importance specified by priority levels
C) Upon each interrupt, a piece of OS codes is called and executed
D) Interrupts cannot be generated by external hardware

3) Which of the following statements is **NOT true** on *process* and *thread*?
A) At a certain moment, only one process is running on a CPU core
B) One process with a single thread can run on different CPU cores at a time in parallel
C) A process can consist of multiple thread(s), and they run within the same address space
D) One process consists of only one address space

TA responsible for HW1: Jingcan Chen (jchenhv@cse.ust.hk)

4) Which of the following statements is **not true** about the goal of *operating system*?
(A)Execute user programs and make solving user problems easier
(B)Allow or deny user's access to hardware resources
(C)Make the computer system convenient to use
(D)Manage and use the computer hardware in an efficient manner

5) Which of the following statements is **true** about *system calls* and *dual mode operation*?
A) Under dual mode operation, both user and the operating system can access all resources
B) The concept of dual modes can only have two modes
C) Some instructions are privileged in system calls
D) Context switch happens during system calls

6) Which of the following statements is **not true** about *loadable kernel module* approach?
A) All components in the kernel can be dynamically added and removed during runtime
B) The kernel has a set of core components and can link in additional services via modules, either at boot time or during run time.
C) Module approach resembles a layered design in that each kernel section has a well-defined, protected interface, and it is flexible to call other modules
D) Loadable modules can add new services to the kernel without recompiling the whole kernel.

7) Which of the following statement is TRUE for *direct memory access* or DMA?
A) DMA transfer data between an I/O device and memory directly without any assistance from CPU
B) DMA completes a data transfer without interrupting CPU
C) DMA frees up CPU from data movement between an I/O device and memory
D) All of the above

8) Which is true about *signal handling* in operating systems?
A) Signals cannot be ignored and should terminate the program since they hurt system performance
B) Keyboard interrupt is a synchronous signal because it happens during the process execution, as the illegal memory access
C) Signals should be sent to every thread in a multi-threaded process
D) Each signal has a default handler function

9) When the degree of multiprogramming is too high, ___
A) Long-term scheduler will remove some processes from the memory
B) Short-term scheduler assigns less CPU cores to a certain process
C) Mid-term scheduler swaps out certain processes to the disk
D) None of the above

TA responsible for HW1: Jingcan Chen (jchenhv@cse.ust.hk)

10) Which is **not true** about *interprocess communication (IPC)*?
A) There are generally two models of IPC, i.e. shared memory and message passing
B) Pipes can be used only in the parent-child processes or among multiple threads of a process
C) Message sending need not block the processes
D) Different processes communicating through sockets must have different port numbers

2. [30 points] Please answer the following questions in a few sentences

(1) (5 points) Please describe what the CPU needs to specify prior to DMA operations, and illustrate why this is better than programmed I/O when moving large chunks of data.

(2) (5 points) Please briefly explain the two essential properties (i.e., *spatial and temporal locality*) why caching works.

(3) (5 points) What resources are shared and not shared in a multi-threaded process? What is the main benefit of using multiple threads instead of multiple processes?

(4) (5 points) What are the advantages of providing *system call APIs* to users in *dual-mode system*? Please explain from user view and system view.

(5) (5 points) What is *orphan* process? What is the problem with an *orphan* process? How does OS handle that?

(6) (5 points) What is *copy-on-write* ? What is the advantage of using *copy-on-write* in fork() implementation in Linux?

3. **(40 points)** Simple C programs on fork(). Suppose all printf() will be followed by fflush(stdout) even if it is not presented in the code. For all the C programs, you can assume that necessary header files are included

1) (5 points) Consider the following code segments:

```
int main()
{
    pid_t pid1;
    pid_t pid2;

    pid1 = fork();
    pid2 = fork();

    printf("pid1:%d, pid2:%d\n", pid1, pid2);
}
```

(a) How many processes are there in total when this code finishes? Give the answer.

(b) If one process prints "pid1:51234, pid2: 51235", one process prints "pid1:0, pid2: 51236", write down other processes' pid and their outputs.

2) (15 points) Consider the following code segments:

```
int main(){
   int i = 0;
   int cnt = 10;
   for (; i<3; i++){
        pid_t pid = fork();
        if (pid == 0)
            printf("%d\n", cnt);
        else
            cnt += 10;
   }
   return 0;
}
```

(a) Who will print the "cnt", the child process or parent process? Why? How many times of "printf" will this code execute? Give the answer with explanation.

(b) Theoretically, based on Linux fork() mechanism introduced in the lecture, how many memory copies of the variable "cnt" will there be? Explain your answer.

(c) Directly running this piece of code may produce different outputs every time. Briefly explain why.

3) (10 points) Consider the following code segments:

```
#include <stdio.h>
#include <unistd.h>
int main() {
   for (int i = 0; i < 2; ++i) {
      if ((fork() && fork()) || !fork()) {
         printf("A");
         fflush(stdout);
       }
   }
}
```

(a) Determine the total number of "A"s output by the given program, assuming it operates normally. Provide a brief explanation for your answer.

4) (10 points) Fill in the missing blanks using "printf" and "wait" functions, so that the following program will always display the following output:
Output: CDBA

Question:

```
int main() {
    if ( fork() ) {
        wait(0);
        if ( fork() ) {
            BLANK1;
            BLANK2;
            fflush(stdout);
        } else {
            printf("B");
            fflush(stdout);
        }
    } else {
        if ( !fork() ) {
            BLANK3;
            fflush(stdout);
        } else {
            BLANK4;
            printf("D");
            fflush(stdout);
        }
    }
    return 0;
}
```

BLANK1	
BLANK2	
BLANK3	
BLANK4	