# Spring 2022 COMP 3511 Homework Assignment 1 (HW1)

**Handout Date: Feb. 18, 2022, Due Date: Mar. 4, 2022**

| | |
|---|---|
| Name | **Solution** |
| Student ID | |
| ITSC email | @connect.ust.hk |

**Please read the following instructions carefully before answering the questions:**

- You should finish the homework assignment **individually**.
- This homework assignment contains **three** parts:
  **1) Multiple choices 2) Short Answer 3) Program with fork()**
- **Homework Submission:** submit to **Homework #1** on **Canvas**.
- TA responsible for HW1: Shuowei (scaiak@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 |
|---|---|---|---|---|---|---|---|---|---|
| C | B | B | D | A | A | C | A | C | D |

1) Which of the following items is not a component of hardware?

A) CPU
B) Basic computing resources
C) Database
D) I/O devices

Answer: C
Explanation:
Database belongs to application programs. See 1.20

2) Which of the following sentences about interrupt is incorrect?

A) Interrupts are widely used in modern operating systems to handle asynchronous events.
B) Some modern operating systems (e.g., Windows 10) are not interrupt-driven.
C) Interrupts have different priority. Low-priority interrupts' execution can be preempted by a high-priority interrupt.
D) I/O device can trigger interrupts by sending a signal to the CPU.

Answer: B
Explanation:
A: See 1.28.
B: See 1.31. All modern OS are interrupt-driven.
C: See 1.31.
D: See 1.30.

3) Which of the following sentences about storage is <u>incorrect</u>?

A) In a hierarchy storage system, fast access of a kind of storage often means a smaller storage capacity.
B) Two essential design issues for cache memory are cache size and access privileges.
C) Mechanical storage is generally larger and less expensive per byte than electrical storage. It also needs more access time.
D) The access time of the register is faster than that of the cache.

Answer: B
Explanation:
A: See 1.33.
B: See 1.36. The replacement policy and cache size are two important design issues of cache memory.
C: See 1.35.
D: See 1.38

4) Which of the following items is <u>NOT</u> an advantage of virtualization?

A) It allows operating systems to run as applications within another operating system.
B) It allows a single physical machine can run multiple operating systems concurrently.
C) It allows multiple users to run tasks concurrently in a system.
D) It allows programs to use the same resource (like storage) at the same time.

Answer: D
Explanation:
A: See 1.48.
B: See 1.48.
C: See 1.51.
D: Virtualization is unable to do this. Take storage as an example, virtualization cannot enable two programs to write there data in a same block, because they will overwrite the data of the other.

5) Command Line Interface (CLI) ____________

A) allows users to enter commands to be performed by the operating system directly.
B) is always implemented in kernel.

C) uses the same syntax in different operating systems.
D) is not applicable when Graphical User Interface (GUI) is used.
Answer: A
Explanation:
See 2.8.
B: CLI can be implemented by system programs.
C & D: common sense.

6) _____ provide(s) an interface to the services provided by an operating system.

A) System calls
B) Application Program Interface (API)
C) Command line Interface
D) Graphical User Interface (GUI)

Answer: A
Explanation:
See 2.14. System calls are the programming interface to the services provided by the OS.

7) Which of the following sentences about system call is incorrect?

A) It is preferred to use APIs rather than directly invoking system calls
B) System calls are generally available as functions written in a high-level language and assembly languages
C) The parameters of system call can only be passed by using registers.
D) A system call interface is provided by the run-time environment for most programming languages.

Answer: C
Explanation:
A: See 2.17.
B: See 2.14.
C: See 2.20. There are three methods for passing parameters to the OS.
C: See 2.18.

8) _______ is used to identify legitimate users of a system.

A) User authentication
B) Application Program Interface (API)
C) Graphical User Interface (GUI)
D) Access right

Answer: A
Explanation:

See 2.36.

9) In _______, messages must be copied between the services when two user-level services must communicate with each other.

A) Monolithic structure
B) Layered Structure
C) Microkernel Structure
D) Loadable kernel modules Structure

Answer: C
Explanation:
See 2.50.

10) What is true about the return value of a successful fork()?

A) fork() will return 0 at the parent process.
B) fork() will return the child's process ID at the child process.
C) fork() will return -1 at parent process.
D) fork() will return the child's process ID at the parent process.

Answer: D
Explanation:
fork() will return the child's process ID at the parent process.
fork() will return 0 at the child process.
fork() will return -1 when it is failed. This question is asking about the return value of a successful fork() function call. So, the answer C is not true
See 3.30.

**2. (30 points) Short answer**

(1) (6 points) Please illustrate the jobs of operating systems in two aspects. When answering this question, please refer to the Operating System Definition slides

Answer:
OS allocate resources. It manages hardware and software. It also makes decisions between conflicting requests for efficient and fair resource use (3 points).
OS control the execution of programs. It prevents errors and improper use of the computer (3 points).
See 1.23.

If you do not refer to the Operating System Definition slides and illustrate the jobs in user & system aspects, you may loss some points.

(2) (6 points) Briefly summarize the advantages of multi-core single-processor systems.

Answer:
Use significantly less power
It is more efficient as internal on-chip communication is faster than communication between different processors.
Good multitasking compared to single core single processor systems.
…
See 1.44

Two correct answer is enough to get the full score of this question.

(3) (6 points) Please contrast the monolithic kernel design with the microkernel approach in terms of efficiency, scalability, and stability.

Answer:
Efficiency: Monolithic kernel design is faster in execution than microkernel design because there is little overhead in the system-call interface and communication within the kernel in microkernel design (2 points). See 2.43.

It is easier to extend a microkernel OS, as all new services can be added to user space without modification on the kernel. However, in monolithic approach, the kernel becomes large and difficult to manage when it needs to be extended (2 points). See 2.48 & 2.50.

One drawback of the monolithic approach is that an enormous number of functionalities are combined to one level, making it difficult to implement and maintain. If a service crashes, the whole system crashes in a monolithic kernel. However, in microkernel approaches, the crash of service has no effects on the whole system. (2 points). See 2.43.

(4) (6 points) Please explain the relation between Linker and Loaders.

Answer: Source codes are compiled into object files designed to be loaded into any physical memory location. Linker is the tool to combine objects into a binary executable file (2 points). The binary executable file is brought into memory by Loader for execution (2 points). A source code is transformed into a binary executable file by Linker and brought into memory for execution by Loader. The sources code will be processed by Linker at first and then Loader for execution (2 points). See 2.33.

(5) (6 points) Describe the similarities and differences between the two processes after calling fork().

**Answer**:
Similarities: Address spaces, variables, process resources, working directory… (2 answers is enough.)
Different: Return value process ID, running time… (2 answers is enough. But you will lose 1 point if your answers do not include the return value ID)

## 3. (40 points) Simple C programs on fork()

For all the C programs, you can assume that necessary header files are included

1) (10 points) Consider the following code segments:

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

How many '30 ' will this code print? Please elaborate.

**Answer: 4** (5 points).
Explanation: The code print '30' at the second iteration of the program. 2 processes print at the first iteration. 2*2 process print at the second iteration, 2*2*2 process print at the third iteration. Therefore, the answer is 4.

How many '+' will this code print? Please elaborate.
**Answer: 14** (5 points).
Explanation: The code print '+' in each iteration of the program. 2 processes print at the first iteration. 2*2 process print at the second iteration, 2*2*2 process print at the third iteration. Therefore, the answer is 2+4+8=14.

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

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

How many times will this code print none zero process ID (pid)? Please elaborate.

**Answer: 2** (5 points).

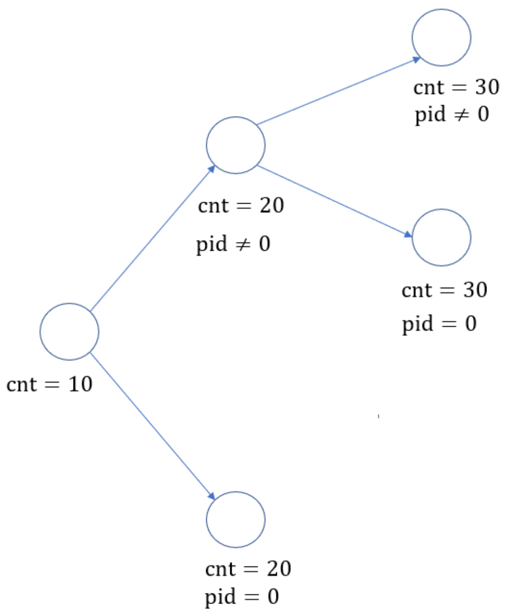
How many '0' will this code print? Please elaborate.

**Answer: 2** (5 points). If you take 'cnt' into consideration, there will be 6 '0'.

How many '30' will this code print? Please elaborate.

**Answer: 2** (5 points).

Explanation: The result of the program can be shown as the figure below.



3) (5points) Consider the following code segments:

```
int main() {
    pid_t pid = fork();
    printf("%d \n",getpid());
    printf("%d \n",pid);
    return 0;
}
```

Function description:

```
getpid(); //get current process id
```

If one process prints:

```
54319
54320
```

What will another process print? Please elaborate.

**Answer: 54320 0 (2.5 points for each correct answer).**

Explanation: The process does not print 0 at 'printf("%d \n",pid)'. Therefore, it is the parent process, and 54320 is the child's process ID. As a result, the child process will print its process ID 54320 and the return value of fork(), which is 0.

See 3.30

4) (10 points) Consider the following code segments:

```
int main() {
    int i = 0;
    for (; i<2; i++) {
        if(______X_______) {
            fork();
            printf("(A,%d) \n",i);
        }
        else {
             printf("(B,%d) \n",i);
        }
    }
    return 0;
}
```

(a) If the code always prints '(B,0)' twice, what is the possible code at position **X**? **X** contains no more than 20 characters

You can assume that fork() is always successful. You MUST use fork() in your answer. Please note that the program prints other lines not equal to (B,0). Please elaborate.

**Answer: fork()<0, fork()&&fork()… (5 points, There are other answers).**

Explanation: '(B,0)' is printed twice, which means two processes go into the 'else' part of the program at the first iteration. Because X is the condition, its value must ensure that two process goes into the 'else' part. Therefore, 'fork()<0' is a feasible solution.

(b) Assume the child's PID created by fork() is always greater than 1000. If the code always prints '(B,1)' for six times, but prints '(B,0)' for only once, what is the possible code at position **X**? **X** contains no more than 20 characters.

You can assume that fork() is always successful. You MUST use fork() in your answer. Please note that the program prints other lines not equal to (B,1) and (B,0). Please elaborate.

**Answer: fork()==i, fork()>0 && i==0 …(5 points, There are other answers).**

Explanation: One process goes into the 'else' part of the program at the first iteration, and six processes go into the 'else' part of the program at the second iteration. The significant difference in the first and the second iteration may be caused by the changing conditions in different iterations.

If there is only one fork() in X, and only one process goes into the else part in the first iteration, then at the end of X in the next iteration, there will be six process. If they all go into the else part, there will be one '(B,0)' and six '(B,1)'.Therefore, combining 'i' and 'fork()' in **X** together is a possible way. 'fork()==i' is a feasible solution.