TA responsible for HW1: Xiaodian Cheng (xchengaq@connect.ust.hk)

Spring 2023 COMP 3511 Homework Assignment 1 (HW1)

Handout Date: Feb. 16, 2023. Due Date: Mar. 2, 2023

Name	ZHAO Yu Xuan
Student ID	20497819
ITSC email	yx2hao@connect.ust.hk @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) Simple C programs on fork()
- **Homework Submission:** submit to **Homework #1** on **Canvas**.
- TA responsible for HW1: Xiaodian Cheng (xchengaq@connect.ust.hk)

1. (30 points) Multiple choices

Write your answers in the boxes below:

MC1	MC2	MC3	MC4	MC5	MC6	MC7	MC8	MC9	MC10

B 1) Which of the following statements is correct?
A) Operating systems include compilers for higher-level languages.
B) Mobile operating systems often include middleware to provide additional services.
C) I/O devices are essential components of the CPU in Von Neumann Architecture.
D) CPU is heavily involved in the entire process of DMA data transfer to improve transmission efficiency.

D 2) Which of the following statements is correct?
A) Main memory is categorized into non-volatile secondary storage in the storage system, which the CPU can directly access.
B) Larger cache hit ratio usually leads to a longer average access time.
C) In a symmetric multiprocessing system, the master processor often assigns specific tasks to the slave processors, and the master handles I/O.
D) Different operating systems can run concurrently on a single physical machine with virtual machines.

D 3) When the block method is used to pass the parameters of system calls to the operating system, _____.

A) the detailed contents of all parameters are stored in registers

B) parameters are pushed onto a stack by the program

C) the number of registers limits the number and length of parameters

D) the address of the block where parameters are stored is passed via the register

C 4) Before running a program on a CPU core, the executable file is brought into memory by _____.

A) assembler

B) linker → executable file

C) loader

D) compiler → object file

C 5) Which of the following statements about operating system structure is <u>correct</u>?

A) Windows XP is designed with the layered approach. → hybrid

B) In the loadable kernel module approach, each layer invokes operations on lower-level layers, simplifying the construction and debugging.

C) It is easy to add new services to the microkernel-based operating system without modifying the kernel, greatly improving the performance of microkernels.

D) Microkernel provides communication between the client program and the various services running in user space through message passing.

A 6) Considering the memory layout of the following C program, the pointer `*x` (in the red dashed box) is stored in _____.

```
#include<stdio.h>
int *x;
int main(int argc, char *argv[])
{
    x = (int *)malloc(sizeof(int)*5);
    free(x);
    return 0;
}
```

A) uninitialized data section

B) initialized data section

C) stack

D) heap

TA responsible for HW1: Xiaodian Cheng (xchengaq@connect.ust.hk)

A 7) After the completion of one I/O operation requested by the process, the process state will be changed _____.

A) from waiting to ready
B) from running to ready
C) from running to waiting
D) from ready to running

B 8) In which of the following scenarios does the system switch from user mode to kernel mode?
I. Divide an integer by zero.
II. Access the data in a file with the `read()` system call.

A) Only I.
B) Only II.
C) Both I and II.
D) Neither of the two scenarios.

9) Cooperating processes cannot use _____ to communicate.

A) shared memory
B) message passing
C) global variables in the C program
D) pipes

C 10) Which of the following statements about the parent process and child process is <u>correct</u>?

A) In UNIX, `fork()` creates a new process, which duplicates the same pid of the parent process.
B) Parent process and children processes have different process control blocks.
C) Parent process can create children processes but cannot terminate its children process.
D) If the parent process is terminated without invoking `wait()`, the child process becomes a zombie process.
orphan

TA responsible for HW1: Xiaodian Cheng (xchengaq@connect.ust.hk)

2. (30 points) Short answer

(1) (6 points) What are the main goals of the operating system? Please answer from the user view and the system view, respectively.

User view:

The goal of the OS is to provide an intermediary program that could be convenient, easy to use, good in performance, and secure to execute user program between user application and computer hardware.

System view:

OS serves as a resource allocator and a control program.

(2) (6 points) Please illustrate two advantages of using APIs rather than invoking system calls directly.

- Using API could increase the program portability which means a designed program could be run and compile on any system that supports the same API.

- API could help user hide the complex details

(3) (6 points) Please illustrate the advantages of the multiprocessor systems (4 points). What is the possible reason for the problem that, compared to a computer with a single processor, the speed-up ratio of a computer with eight identical processors is less than eight (2 points)?

The multiprocessor system could increase throughput that gives more computing capability. And it has the benefit of economy of scale which could share other devices such as I/O device.
Moreover, it could increase the reliability that has graceful degradation or fault tolerance.

Since multiprocess has the problem overhead such as contention for shared resources including bus and memory, this issue will lead to a phenomenon that the speed-up ratio of N processor is less than N.

(4) (6 points) Please summarize the function of the process scheduler in brief (2 points). A common representation of process scheduling is a queueing diagram, including two different types of scheduling queues of processes. Please describe the state of the processes placed in these two types of queues, respectively (4 points).

The process scheduler is an OS mechanism that used to select a process for execution on one CPU core and guarantee the selection of the available process is CPU-utilized, fair, latency-optimized, and the like.

One is the ready queue which stores a set of processes in main memory, ready and waiting to execute.

The other one is the waiting queue which stores a set of processes waiting for an event such as completion of I/O.

TA responsible for HW1: Xiaodian Cheng (xchengaq@connect.ust.hk)

(5) (6 points) Please illustrate the main advantage of dual-mode operation (i.e., why should the system distinguish between user mode and kernel mode) (2 points) and describe the workflow of dual-mode operation after a user application requests a service via a system call (4 points).

Dual-mode operation enables the protection of OS from other system components and it also could protect the user program from the another user program.

the request of a service via a system is called a trap or interrupt and it will switch the mode bit from 1 to zero to activate the kernel mode for some privileged instruction execution.

After the system call returned, the mode bit will be reset to 1 from zero to resume the user mode.

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

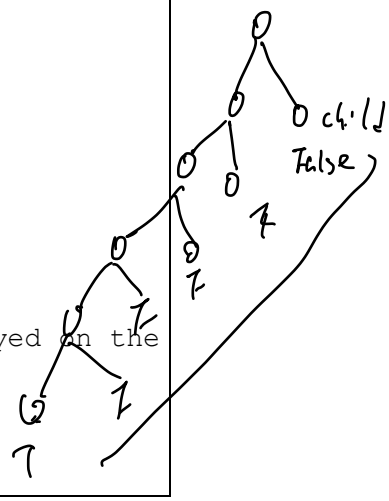
For all the C programs below, you can assume that necessary header files are included and `fork()` always creates a new process successfully.

TA responsible for HW1: Xiaodian Cheng (xchengaq@connect.ust.hk)

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

```
int main(){
    pid_t pid, id;
    for (int i = 0; i < 5; i++){
        id = fork();
        if(id == 0) break;
    }
    pid = getpid();
    if (pid == id) printf("true\n");
    else printf("false\n");
    fflush(stdout); //force the output to be displayed on the
console before the program exits
    return 0;
}
```

child process pid



How many "true" and "false" will be printed, respectively (4 points)? Please elaborate (6 points).

true: 0 time ; false: 6 times .

From the for loop, the parent process will execute 5 time fork() system calls.
Once the child process is created, it will directly break the child process since the fork() system call will return zero if it's in child process.
And there is no process that have pid = 0, hence the five child processes will output five times false.
Then after the parent process finished the execution of for loop, it will also give the false output because the fork() will return the child process's pid which is different from the parent process's pid.

2) (10 points) Consider the following code segments:

```
int main(){
    if (fork() || fork()){
        printf("true\n");
    }
    printf("process\n");
    fflush(stdout);
    return 0;
}
```

How many "true" will be printed (3 points)? How many "process" will be printed (2 points)? Please elaborate (5 points). (Hint: In C language, if State1 is true, "State1 | | State2" will directly return true without checking State2.)

true: 2 times ; process : 3 times

the left fork() will create a parent and child process which the parent process will only output one true and one process.

And the child process will further execute the right fork() since the fork() gives 0 return value for child process. the original child process will only output one process.

the newly created parent process will give one true and process.

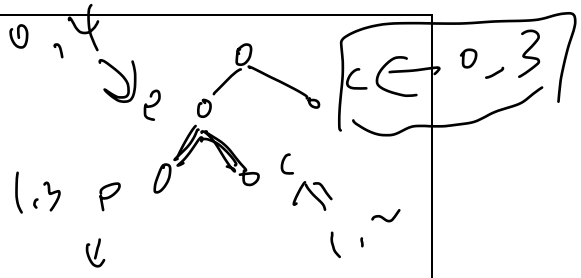
the newly created child process will only output one process.

As a result, there are 2 times true and three times process is printed in total.

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

```
int value = 5;
void recursion(int iteration){
    int pid;
    if (iteration >= 2) return;
    pid = fork();
    value --;
    if (pid){
        wait(NULL);
        printf("%d, %d\n", iteration, value);
        recursion(iteration + 1);
        fflush(stdout);
    }
    else{
        value --;
        printf("%d, %d\n", iteration, value);
        fflush(stdout);
        exit(0);
    }
}

int main(){
    recursion(0);
    return 0;
}
```



What is the output (4 points)? Please elaborate (6 points).

Output: 0, 3
0, 4
1, 2
1, 3

The zeroth iteration will have one parent and one child process by the execution of fork().
The parent process has the values of 0, 4 for iteration and value respectively.
Then the parent process will wait for the termination of the child process.
The child process will firstly print 0, 3 since the value deduct one for two times.
Then the parent process continue to execute the print instruction which gives 0, 4.

And it will recursively call the function with iteration = 1. The similar process will be done like the above elaboration. The child process will first print the values of iteration and value, hence it will output 1, 2. The value is equal to 2 because the orignal value of 4 got deducted by 1 for two times. Lastly, the parent process will print 1, 3. Even though it will recursively call the function, the iteration already reached 2 that will return the function.

4) (10 points) In this question, we introduce the macro `WEXITSTATUS(int status)` defined in <sys/wait.h> header. This macro evaluates to the least significant 8 bits of the exit status value that the child process passed to `exit()`. You can take the following code as an example.

```
int main(){
   int status;
   pid_t pid = fork(); // Create a child process.
   if(pid == 0) exit(50); // The child process terminates and
passes 50 as the exit status value.
   wait(&status); // The parent process waits and receives the
exit status value of the child process.
   int value = WEXITSTATUS(status); // value = 50
   return 0;
}
```

In this question, we want to use fork() to implement a dot product between two vectors with multiple processes. For example, given two vectors $\boldsymbol{a} = [3, 2, 4]$ and $\boldsymbol{b} = [1, 5, 2]$, the dot product between $\boldsymbol{a}$ and $\boldsymbol{b}$ is $\boldsymbol{a} \cdot \boldsymbol{b} = 3 * 1 + 2 * 5 + 4 * 2 = 21$. The input of our program includes two vectors. The output is the dot product between them.

The input data satisfies the following conditions.

1. The elements of the vectors are all non-negative integers.
2. The number of elements in each vector is fixed to 5.
3. The result of the dot product is less than 256.

Please fill in the five blanks to complete the following program. <u>You can write at most 20 characters in each blank</u> (10 points, 2 points for each blank).

TA responsible for HW1: Xiaodian Cheng (xchengaq@connect.ust.hk)

```
#define size 5
int main() {
    int i, status, product, value;
    int a[size], b[size];
    pid_t pid;
    for (i = 0; i < size; i++){
        scanf("%d", &a[i]); // Input vector a
    }
    for (i = 0; i < size; i++){
        scanf("%d", &b[i]); // Input vector b
    }
    for (i = 0; i < size - 1; i++){
        pid = fork(); // Create processes
        if(!pid){
            Blank 1 break;
        }
    }
    product = Blank 2 a[i] * b[i]; // Calculate the product
    if(!pid) exit(product);
    for (i = 0; i < Blank 3 ; i++){
        Blank 4 ;
        value = WEXITSTATUS(status);
        product = Blank 5 ; // Calculate the dot product
    }
    printf("The result of the dot product is %d.\n", product);
    return 0;
}
```

Sample Input:

3 4 3 1 5

2 1 4 5 2

Sample Output:

The result of the dot product is 37

Write your answer in the following blanks.

Blank 1: break

Blank 2: a[i] * b[i]

Blank 3: size - 1

Blank 4: wait(&status)

Blank 5: product + value