Started on	Monday, 5 October 2020, 9:11 AM
State	Finished
Completed on	Monday, 5 October 2020, 9:50 AM
Time taken	38 mins 55 secs
Grade	**11.00** out of 18.00 (**61**%)

Question **1**
Complete
Mark 1.00 out of 1.00

Indicate whether the system call fork() will cause the calling process to enter into waiting state.

Select one:
- True
- [✓ False]

The correct answer is 'False'.

Question **2**
Complete
Mark 1.00 out of 1.00

Indicate whether the system call getpid() will cause the calling process to enter into waiting state.

Select one:
- True
- [✓ False]

The correct answer is 'False'.

Question **3**
Complete
Mark 0.00 out of 1.00

Indicate whether the system call ReadConsole() will cause the calling process to enter into waiting state.

Select one:
- True
- [✓ False]

The correct answer is 'True'.

Question **4**

Complete

Mark 1.00 out of 1.00

Indicate whether the system call wait() will cause the calling process to enter into waiting state.

Select one:

[✓ True]

False

The correct answer is 'True'.

Question **5**

Complete

Mark 1.00 out of 1.00

Indicate whether the system call WriteConsole() will cause the calling process to enter into waiting state.

Select one:

[✓ True]

False

The correct answer is 'True'.

Question **6**

Complete

Mark 1.00 out of 1.00

```
What is a process control block (PCB)?
```

Select one:

a.
```
None of the answers.
none of the above
```

[✓ b.]
```
A data structure that stores all the information about a process that is necessary for execution.
```

c.
```
A critical section within the OS kernel code.
```

d.
```
A special type of software interrupt that stops a process.
```

e.
```
The layer of the OS kernel that is responsible for process management.
```

The correct answer is:
```
A data structure that stores all the information about a process that is necessary for execution.
```

Question **7**
Complete
Mark 1.00 out of 1.00

When an interrupt happens, the hardware saves the context of the running process onto the stack.

The correct answer is: stack

Question **8**
Complete
Mark 2.00 out of 2.00

Suppose we run a program based on the following code. Write down the sequence of the states (i.e., N (NEW), R (RUNNING), W (WAITING), Y (READY), T (TERMINATED) etc) that the process will go through. Assume there is no other process. Use N, R, W, Y, T to represent the states (e.g. NYRT).

```
int main()
{
   foo();
}
int foo()
{
   int c[2];
   int i;
   for(i=0; i<2; ++i)
      scanf("%d\n", c[i]);
   return 0;
}
```

Answer: NYRWYRWYRT

The correct answer is: NYRWYRWYRT

Question **9**
Complete
Mark 1.00 out of 1.00

Indicate for the following statements as to whether they are true.

(b) The statement after a call to exec() will never be executed.

Select one:
True
[✓ False]

The correct answer is 'False'.

Question **10**

Complete

Mark 0.00 out of 1.00

Indicate for the following statements as to whether they are true.

(a) After calling execve, a process has a new processs ID.

Select one:

[✓ True]

False

The correct answer is 'False'.

Question **11**

Complete

Mark 0.00 out of 2.00

```
A binary executable file successfully compiled from the following code.c is named code.exe.

#include <stdio.h>

int main(int argc, char *argv[])

{

    printf("argc = %d argv[0] = %s\n", argc, argv[0]);

}


Suppose we attempt to run a program using the following line of code. Assuming that code.exe can be found on the environment paths,

what is the printout of the process executing this code?

execlp("code", "code", "haha", "hee hee", NULL);
```

Select one:

a.
```
argc = 1 argv[0] = haha
```

b.
```
argc = 2 argv[0] = haha
```

[✓ c.]
```
argc = 4 argv[0] = code
```

d.
```
argc = 3 argv[0] = code
```

e.
```
argc = 2 argv[0] = hee hee
```

The correct answer is:
```
argc = 3 argv[0] = code
```

Question **12**

Complete

Mark 1.00 out of 1.00

Complete a simple program with main function, that creates 2 child processes using fork().

The child process prints "Hello" together with its process ID while the parent prints "World" together with its child process IDs.

Your program should ensure that the print out shall be always "Hello ***PID1*** Hello ***PID2*** World ***PID1 PID2***",

where ***PID1*** and ***PID2*** are respectively the process IDs of the two child processes.

You may assume that all necessary headers are already included.

```
int main(int argc, char *argv[])
{
    pid_t cpid1, cpid2;
    int status;
    cpid1 = fork();
    if ( cpid1==0 ) { /*blank 1*/
        printf("Hello %ld ", (long) ___________);/*blank 2*/
        fflush(stdout);
        _exit(EXIT_SUCCESS);
    } else {
        ______________;/*blank 3*/
        ______________; /*blank 4*/
        if (______________) { /*blank 5*/
            printf("Hello %ld ", (long) ________);/*blank 6*/
            fflush(stdout);
            _exit(EXIT_SUCCESS);
        }
        else {
            _______________;/*blank 7*/
            printf("World %ld %ld\n", cpid1, cpid2);
        }
    }
    exit(EXIT_SUCCESS);
}
```

Complete the answer for blank 1 without extra space.

The correct answer is: cpid1==0

Question **13**

Complete

Mark 0.00 out of 1.00

Complete a simple program with main function, that creates 2 child processes using fork().

The child process prints "Hello" together with its process ID while the parent prints "World" together with its child process IDs.

Your program should ensure that the print out shall be always "Hello ***PID1*** Hello ***PID2*** World ***PID1 PID2***",

where ***PID1*** and ***PID2*** are respectively the process IDs of the two child processes.

You may assume that all necessary headers are already included.

```
int main(int argc, char *argv[])
{
    pid_t cpid1, cpid2;
    int status;
    cpid1 = fork();
    if ( _ _ _ _ _ _ _ _ _ _ _ _ _ ) { /*blank 1*/
        printf("Hello %ld ", (long) [ cpid1 ]);/*blank 2*/
        fflush(stdout);
        _exit(EXIT_SUCCESS);
    } else {
        _______________;/*blank 3*/
        _______________; /*blank 4*/
        if (_______________) { /*blank 5*/
            printf("Hello %ld ", (long) _________);/*blank 6*/
            fflush(stdout);
            _exit(EXIT_SUCCESS);
        }
        else {
            ________________;/*blank 7*/
            printf("World %ld %ld\n", cpid1, cpid2);
        }
    }
    exit(EXIT_SUCCESS);
}
```

Complete the answer for blank 2 without extra space.

The correct answer is: getpid()

Question **14**

Complete

Mark 0.00 out of 1.00

Complete a simple program with main function, that creates 2 child processes using fork().

The child process prints "Hello" together with its process ID while the parent prints "World" together with its child process IDs.

Your program should ensure that the print out shall be always "Hello ***PID1*** Hello ***PID2*** World ***PID1 PID2***",

where ***PID1*** and ***PID2*** are respectively the process IDs of the two child processes.

You may assume that all necessary headers are already included.

```
int main(int argc, char *argv[])
{
    pid_t cpid1, cpid2;
    int status;
    cpid1 = fork();
    if ( _ _ _ _ _ _ _ _ _ _ _ _ _ ) { /*blank 1*/
        printf("Hello %ld ", (long) _ _ _ _ _ _ _ _ _ _ _ _ _);/*blank 2*/
        fflush(stdout);
        _exit(EXIT_SUCCESS);
    } else {
        cpid2=fork() ;/*blank 3*/
        _______________; /*blank 4*/
        if (_______________) { /*blank 5*/
            printf("Hello %ld ", (long) _________);/*blank 6*/
            fflush(stdout);
            _exit(EXIT_SUCCESS);
        }
        else {
            ________________;/*blank 7*/
            printf("World %ld %ld\n", cpid1, cpid2);
        }
    }
    exit(EXIT_SUCCESS);
}
```

Complete the answer for blank 3 without extra space.

The correct answer is: wait(&status)

Question **15**

Complete

Mark 0.00 out of 1.00

Complete a simple program with main function, that creates 2 child processes using fork().

The child process prints "Hello" together with its process ID while the parent prints "World" together with its child process IDs.

Your program should ensure that the print out shall be always "Hello ***PID1*** Hello ***PID2*** World ***PID1 PID2***",

where ***PID1*** and ***PID2*** are respectively the process IDs of the two child processes.

You may assume that all necessary headers are already included.

```
int main(int argc, char *argv[])
{
    pid_t cpid1, cpid2;
    int status;
    cpid1 = fork();
    if ( _ _ _ _ _ _ _ _ _ _ _ _ _ ) { /*blank 1*/
        printf("Hello %ld ", (long) _ _ _ _ _ _ _ _ _ _ _ _ _ );/*blank 2*/
        fflush(stdout);
        _exit(EXIT_SUCCESS);
    } else {
        _ _ _ _ _ _ _ _ _ _ _ _;/*blank 3*/
        wait(&status) ; /*blank 4*/
        if (_______________) { /*blank 5*/
            printf("Hello %ld ", (long) ________);/*blank 6*/
            fflush(stdout);
            _exit(EXIT_SUCCESS);
        }
        else {
            _______________;/*blank 7*/
            printf("World %ld %ld\n", cpid1, cpid2);
        }
    }
    exit(EXIT_SUCCESS);
}
```

Complete the answer for blank 4 without extra space.

The correct answer is: cpid2=fork()

Question **16**

Complete

Mark 1.00 out of 1.00

Complete a simple program with main function, that creates 2 child processes using fork().

The child process prints "Hello" together with its process ID while the parent prints "World" together with its child process IDs.

Your program should ensure that the print out shall be always "Hello ***PID1*** Hello ***PID2*** World ***PID1 PID2***",

where ***PID1*** and ***PID2*** are respectively the process IDs of the two child processes.

You may assume that all necessary headers are already included.

```
int main(int argc, char *argv[])
{
    pid_t cpid1, cpid2;
    int status;
    cpid1 = fork();
    if ( _ _ _ _ _ _ _ _ _ _ _ _ _ ) { /*blank 1*/
        printf("Hello %ld ", (long) _ _ _ _ _ _ _ _ _ _ _ _ );/*blank 2*/
        fflush(stdout);
        _exit(EXIT_SUCCESS);
    } else {
        _ _ _ _ _ _ _ _ _ _ _ _ _;/*blank 3*/
        _ _ _ _ _ _ _ _ _ _ _ _ _; /*blank 4*/
        if ( cpid2==0 ) { /*blank 5*/
            printf("Hello %ld ", (long) ________);/*blank 6*/
            fflush(stdout);
            _exit(EXIT_SUCCESS);
        }
        else {
            _______________;/*blank 7*/
            printf("World %ld %ld\n", cpid1, cpid2);
        }
    }
    exit(EXIT_SUCCESS);
}
```

Complete the answer for blank 5 without extra space.

The correct answer is: cpid2==0

◄ Assignment 2 Part 2

Jump to...

Quiz 3 ►