Your grade: 100%

Your latest: **100%** • Your highest: **100%** • We keep your latest score.

Next item →

Instructions

1. Consider the C program snippet shown below. Assume all relevant header files are included, all function calls execute successfully and the code compiles successfully. Also, assume that after compilation, the execution will be carried out on an Ubuntu system capable of running the code successfully. So, you need to answer the associated question keeping the above-mentioned points in mind. 2 / 2 points

```
int main(){
    for(int i = 0; i < 5; i++)
            fork();
    return 0;
}
```

How many processes in total are created when the above code is executed? Make sure you count every single process that gets created.

- ○ 10
- ○ 25
- ○ 5
- ◉ 32

Correct
This answer is correct. Let us call the first process that gets created when the code starts execution as P. For i = 0, P creates a child C1. For i = 1, both P and C1 call fork() creating two children processes, C2 and C3. For i = 2, each of P, C1, C2 and C3 call fork() creating 4 more children processes, C4, C5, C6 and C7. For i = 3, each of P to C7, calls fork() creating 8 more children processes, C8, C9, C10, C11, C12, C13, C14 and C15. Finally, for i = 4, each of the 16 processes (P to C15) calls fork(), creating 16 more children processes, C16, C17, C18,….., C31. Hence, a total of 32 processes are created on execution of the code.

2. Consider the C program snippet shown below. Assume all relevant header files are included, all function calls execute successfully and the code compiles successfully. Also, assume that after compilation, the execution will be carried out on an Ubuntu system capable of running the code successfully. So, you need to answer the associated question keeping the above-mentioned points in mind. 2 / 2 points

```
int main(){
    sleep(5); //sleep() function ensures that the calling process sleeps for 5 seconds
    if(fork())
        if(fork())
            if(fork())
                if(fork())
                    if(fork())
                        printf("\n%d\n", getpid());
    return 0;
}
```

Which of the following is true? For the options given below, the meaning of the statement process x creates a child process y is that y is the direct child of x.

- ○ When the code is executed, first the parent process P is created. Then P creates a child process C1. Then C1 creates a child process C2. Then C2 creates a child process C3. Then C3 creates a child process C4. Finally, C4 creates a child process C5. Finally, the PID of C5 is printed.
- ○ When the code is executed, first the parent process P is created. Then P creates a child process C1. Then P creates another child process C2. Then P creates another child process C3. Then C1 creates a child process C4. Finally, C2 creates a child process C5. Finally, the PID of C1 is printed.
- ○ When the code is executed, first the parent process P is created. Then P creates a child process C1. Then C1 creates a child process C2. Then P creates another child process C3. Then C3 creates a child process C4. Finally, C1 creates another child process C5. Finally, the PID of C3 is printed.
- ◉ When the code is executed, first the parent process P is created. Then P creates a child process C1. Then P creates another child process C2. Then P creates a third child process C3. Then P creates a fourth child process C4. Finally, P creates the fifth child process C5. Finally, the PID of P is printed.

Correct
This answer is correct. When P executes the 1st fork() inside the 1st if, C1 is created. The if condition

This answer is correct. When P executes the 1st fork() inside the 1st if, C1 is created. The if condition evaluates to true for P. So, P executes the 2nd fork() inside the 2nd if and creates C2. Again, this if condition evaluates to true for P and P executes the 3rd fork() inside the 3rd if condition and creates C3. Again, this if condition evaluates to true for P and P executes the 4th fork() inside the 4th if condition and creates C4. Finally, P executes the 5th fork() inside the 5th if condition and creates C5. For the 5th if condition, the condition evaluates to true for P and hence the PID of P is printed. Note that all of C1, C2, C3, C4 and C5 are direct children processes of P and no grandchild process is created in this code.

3. Consider the C program snippet shown below. Assume all relevant header files are included, all function calls execute successfully and the code compiles successfully. Also, assume that after compilation, the execution will be carried out on an Ubuntu system capable of running the code successfully. So, you need to answer the associated question keeping the above-mentioned points in mind. **1 / 1 point**

```
int main(){
    pid_t p1, p2;
    p1 = fork();
    if(p1 == 0){
        p2 = fork();
        if(p2 == 0){
            printf("\n1\n");
            return 0;
        }
        wait(NULL);
        printf("\n3\n");
        return 0;
    }
    else{
        wait(NULL);
        printf("\n2\n");
        return 0;
    }
}
```

What output will be produced on executing this code? You can ignore the extra newlines printed in the output while determining which option is correct.

- ○ 1 2 3
- ○ 2 1 3
- ○ 3 2 1
- ◉ 1 3 2

Correct
This answer is correct. Let P be the process that is created on executing the code before any fork() is invoked. When P executes p1 = fork(), a child process C1 is created. If(p1 == 0) evaluates to true for C1. Then, C1 executes p2 = fork() and creates a child process C2. Note that C2 is a grandchild of P. if(p2 == 0) evaluates to true for C2. C2 prints 1 and terminates because of return 0. C1 waits for C2 to terminate and then prints 3 and terminates. The else block evaluates to true for P. P waits for C1 to finish its execution, prints 2 and terminates.

4. Consider the C program snippet shown below. Assume all relevant header files are included, all function calls execute successfully and the code compiles successfully. Also, assume that after compilation, the execution will be carried out on an Ubuntu system capable of running the code successfully. So, you need to answer the associated question keeping the above-mentioned points in mind. **1 / 1 point**

```
int main(){
```

```
    int num = 5;
    if(fork()){
        wait(NULL);
        printf("\nnum = %d\n", num);
    }
    else{
        num++;
        printf("\nnum = %d\n", num);
    }
    return 0;
}
```

Which of the following is true regarding the output on executing this program?

- ○ Parent process prints 6 as the value of num and the child process prints 6 as the value of num.
- ◉ Parent process prints 5 as the value of num and the child process prints 6 as the value of num.
- ○ Parent process prints 5 as the value of num and the child process prints 5 as the value of num.
- ○ Parent process prints 6 as the value of num and the child process prints 5 as the value of num.

Correct
This answer is correct. The if condition execution creates a child process. Now, the if condition evaluates to true for the parent process. Hence, parent waits for the child process, prints 5 as the value of num. Now, the child process executes the else block. The child inherits the variable num from the parent but has its own copy of num. So, when child performs num++, that does not affect the parent's copy of num. Hence, for the child process, num becomes 6 and it prints 6 as the value of num.

5. Consider the C program snippet shown below. Assume all relevant header files are included, all function calls execute successfully and the code compiles successfully. Also, assume that after compilation, the execution will be carried out on an Ubuntu system capable of running the code successfully. So, you need to answer the associated question keeping the above-mentioned points in mind. **2 / 2 points**

```
int main(){
    int a = 15;
    if(fork()== 0){
        int a = 20;
        int b = 30;
        if(fork() == 0){
            b = b + 10;
            Line x: print the values of a and b
        }
        else{
            wait(NULL);
            Line y: print the values of a and b
        }
    }
    else{
        int b = 5;
        b = b * b;
        wait(NULL);
        Line z: print the values of a and b
    }
    return 0;
}
```

What are the values of a and b printed corresponding to Lines x, y and z? Assume that lines x, y and z are actual executable statements and appropriate values of a and b will be printed as a result of their execution.

- ○ Line x prints the value of a as 20 and the value of b as 30, line y prints the value of a as 20 and the value of b as 40 and line z prints the value of a as 15 and the value of b as 30.
- ◉ Line x prints the value of a as 20 and the value of b as 40, line y prints the value of a as 20 and the value

of b as 30 and line z prints the value of a as 15 and the value of b as 25.

- ○ Line x prints the value of a as 15 and the value of b as 40, line y prints the value of a as 20 and the value of b as 30 and line z prints the value of a as 15 and the value of b as 900.
- ○ Line x prints the value of a as 15 and the value of b as 30, line y prints the value of a as 20 and the value of b as 40 and line z prints the value of a as 15 and the value of b as 1600.

Correct

This answer is correct. Here, let P be the initial parent process. P creates a child process C1 and C1 creates a grandchild process (of P) C2. The value of a is 15 for P. The 1st if(fork() == 0) evaluates to true for C1. Now, C1 redeclares a and assigns it a value of 20. This redeclaration of a masks the variable a that C1 inherited from P. So, for C1, a = 20. C1 declares another variable b and assigns it a value of 30. Then, C1 creates C2. The 2nd if(fork() == 0) evaluates to true for C2. C2 inherits a and b from C1. Note C2 does not inherit the a variable from P. Moreover, C2 modifies its copy of b. Thus, in line x, C2 prints a = 20 and b = 40. The else block corresponding to the 2nd if evaluates to true for C1. So, C1 waits for C2 to terminate and then in line y, prints a as 20 and b as 30 since C1's copy of b has not been modified. The else block corresponding to the 1st if (outer if) evaluates to true for P. P declares b and assigns it a value of 5. This b is not related to the b of C1 or C2. Then, P changes the value of b to 25. After this, P waits for C1 to finish its execution. Finally, in line z, P prints a as 15 and b as 25.

6. Consider the C program snippet shown below. Assume all relevant header files are included, all function calls execute successfully and the code compiles successfully. Also, assume that after compilation, the execution will be carried out on an Ubuntu system capable of running the code successfully. So, you need to answer the associated question keeping the above-mentioned points in mind. **2 / 2 points**

```
int main(){
    if(fork()){
        if(fork()){
            if(fork()){
                execlp("/bin/ls", "ls", NULL);
            }
            if(fork()){
                execlp("/bin/echo", "echo", "HELLO", NULL);
            }
        }
    }
    return 0;
}
```

Now, this program is executed. Consider the process created on executing this program (before any fork() is executed) as P. Which of the following statements is true?

- ○ Two direct children of P, say C1 and C2 and two grandchildren of P, say C3 and C4 are created.
- ○ One direct child of P, say C1 is created and three grandchildren of C1, say C2, C3 and C4 are created.
- ◉ Three direct children of P, say C1, C2, and C3 and one grandchild of P, say C4 are created.
- ○ One direct child of P, say C1 is created, one child of C1 (grandchild of P), say C2 is created. One child of C2 (grandchild of C1), say C3 is created. Finally, one child of C3 (grandchild of C2), say C4 is created.

Correct

This answer is correct. The fork() inside the 1st if condition creates a direct child of P (C1) and the if condition evaluates to true for P. The fork() inside the 2nd if condition is executed by P which creates another direct child (C2) of P. This 2nd if condition also evaluates to true for P. The fork() inside the 3rd if condition is executed by P which creates the 3rd direct child (C3) of P. Now, this if condition also evaluates to true for P. P executes execlp() and the entire process image of P is erased. The fork() inside the 4th if condition is executed by a direct child of P and this results in the creation of 1 grandchild process of P. Thus, 3 direct children and 1 grandchild of P are created.