System and Device Programming
On Off Exam
20.06.2023

Ex 1 (1.5 points)

Suppose the following program is run using the command:

```
./pgrm 2
```

Indicate a possible program output.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>

int main (int argc, char *argv[]) {
  int i, n;
  char str[40];

  n = atoi (argv[1]);

  for (i=1; i<=n; i++) {
    if ( fork() == 0) {
      sprintf (str, "%d", n-1);
      execlp (argv[0], argv[0], str, NULL);
    }
  }

  printf ("%d", n);
  fflush (stdout);

  exit (0);
}
```

Choose one or more options:

1. ✅ 21100
2. ⬜ 221100
3. ✅ 01012
4. ⬜ 210210
5. ⬜ 110022
6. ✅ 11200
7. ⬜ 2110

Ex 2 (1.5 points)

Indicate the possible output, or outputs, that can be obtained by concurrently executing the following processes PA, PB, and PC with the reported semaphore initialization.

```
init (S1, 0); init(S2, 1);
```

PA	***PB***	***PC***

`wait(S1);`
`printf(“A”);`
	`wait(S2);`
	`printf(“D”);`
		`wait(S2);`
		`printf(“E”);`

```
 signal(S2);                  signal(S1);                  signal(S1);
 wait(S2);
 printf("B");
 wait(S1);
 printf("C");
```

Choose one or more options:

1. ⬜ DABC
2. ⬜ DABCE
3. ⬜ EABC
4. ✅ DAE
5. ⬜ EABCD
6. ⬜ DAD
7. ⬜ EAE
8. ⬜ DAEBC
9. ✅ EAD
10. ⬜ EADBC

Ex 3 (1.5 points)

Given three processes PA, PB, PC, and PD, whose code is reported in the following and whose pids are pid_PA, pid_PB, pid_PC, and pid_PD, respectively. Indicate which of the following outputs is correct. Assume that the `other_code()` function contains neither calls to other blocking functions nor calls to the `kill()` system call. Note that incorrect answers imply a penalty in the final score.

```
PA
kill(pid_PB, SIG…);
pause();
printf("A");

PB
other_code();
pause();
printf("B");
kill(pid_PC, SIG…);
kill(pid_PD, SIG…);

PC
other_code();
pause();
printf("C");

PD
other_code();
pause();
printf("D");
```

Choose one or more options:

1. ✅ No output.
2. ⬜ BACD
3. ⬜ BADC
4. ✅ B
5. ⬜ ABDC
6. ⬜ ABCD

7. ✅ BC
8. ✅ BD
9. ✅ BCD

Ex 4 (1.5 points)

Analyze the following code snippet in C++. When the main is executed, indicate how many (standard) constructors and destructors are called.

```
class C {
   private:
      ...
   public:
     ...
};

void f1(C e) { ... }
void f2(C &e) { ... }

int main() {
   C e1, e2;
   f1(e1);
   f2(e2);
   C *e3 = new C;
   return 0;
}
```

Choose one or more options:

1. ⬜ 2 constructors and 3 destructors.
2. ⬜ 3 constructors and 2 destructors.
3. ⬜ 2 constructors and 2 destructors.
4. ✅ 3 constructors and 3 destructors.
5. ⬜ 4 constructors and 4 destructors.
6. ⬜ 3 constructors and 4 destructors.

Ex 5 (1.5 points)

Analyze the following code snippet in C++. Indicate the possible output or outputs that can be obtained by executing the program.

```
int main() {
    int v1 = 10;
    auto lambda = [v1] (int n1, int &n2) {
        return v1 + n1*n2;
    };
    int i = 2;
    int v2 = lambda(7, i);
    cout << v2;
    return 0;
}
```

Choose one or more options:

1. ⬜ The program displays the value 14
2. ⬜ The program displays the value 20
3. ⬜ The program does not run as there is a bug.
4. ✅ The program displays the value 24
5. ⬜ The variables n1 and n2 in the lambda function are not defined.

6. ⬜ The lambda function must be defined outside the main.

Ex 6 (1.5 points)

Analyze the following code snippet. Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

```
template <typename T, typename R>
R compare(const T& v1, const T& v2) {
   if (v1 < v2) { return -1; }
   if (v2 < v1) { return 1; }
   return 0;
}
```

Choose one or more options:

1. ⬜ The code represents a class template and it allows the code to be instantiated with different types for objects T and R.
2. ⬜ The following function call is correct: `bool v = compare (13.5, 17.6);`
3. ✅ The following function call is correct: `int v = compare (13.5, 17.6);`
4. ✅ The code represents a function template and it allows the code to be instantiated with different types for objects T and R.
5. ⬜ The following function call is correct: `int v = compare (&i, &j);`
6. ⬜ The following function call is correct: `bool v = compare (*i, *j);`

Ex 7 (1.0 points)

Analyze the following code snippet. Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

Code in Thread 1

```
pthread_mutex_lock (&m);
done = 1;
pthread_cond_signal (&cv);
pthread_mutex_unlock (&m);
```

Code in Thread 2

```
pthread_mutex_lock (&m);
while (done == 0)
   pthread_cond_wait (&cv, &m);
pthread_mutex_unlock(&m);
```

Choose one or more options:

1. ✅ The function `npthread_condi_signal` can be substiturted by `pthread_cond_broadcast`.
2. ⬜ If Thread 2 runs before Thread 1, Thread 2 will stop on function `pthread_cond_wait`, but it will not release the mutex `m`.
3. ✅ If Thread 2 runs before Thread 1, Thread 2 will stop on function `pthread_cond_wait`, and it will release the mutex `m`.
4. ⬜ The cycle `while(done==0)` can be substituted by `if(done==0)`.
5. ⬜ If Thread 1 runs before Thread 2, Thread 2 will not stop, but it will execute the function `pthread_cond_wait`.

Ex 8 (1.0 points)

Considering C++ programming with tasks, promises, and futures, indicate which of the following statements is correct. Note that wrong answers imply a penalty in the final score.

Choose one or more options:

1. ⬜ A future is an object that can store a value to be retrieved by a future object.
2. ✅ Each task has an asynchronous policy associated with it; the policy can be `launch::asynch`, `launch::deferred` or the default one.
3. ✅ A future is an object that can represent a value generated by some provider.
4. ⬜ A promise is an object that can represent a value generated by some provider.
5. ✅ A promise is an object that can store a value to be retrieved by a future object.
6. ⬜ Promises are stored in the producer of the promise.

Ex 9 (1.0 points)

Considering IO multiplexing in C language, indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

Choose one or more options:

1. ✅ We can implement IO multiplexing with standard non-blocking IO system calls.
2. ⬜ We can implement IO multiplexing with standard blocking IO system calls.
3. ✅ We can implement IO multiplexing with standard asynchronous IO system calls.
4. ✅ We can implement IO multiplexing using the `select` system calls.
5. ⬜ We can implement IO multiplexing with the system call `mmap` and `munmap`.

Ex 10 (1.0 points)

Indicate which one of the following considerations is correct. Note that more than one response can indeed be correct and that incorrect answers may imply a penalty on the final score.

Choose one or more options:

1. ⬜ Both an ASCII file and a BINARY file can be manipulated with functions fscanf and fprintf.
2. ⬜ An ASCII file is generally more compact than a BINARY one.
3. ⬜ Functions `fopen` and `open` return the same type of object.
4. ✅ An ASCII file is generally more compact than a UNICODE one.
5. ✅ Both an ASCII file and a BINARY file can be read with function read.
6. ⬜ In a binary file, it is impossible to store integer values.
7. ⬜ In C++, the output operator "<<" (e.g, "`cout << value`") is always buffered and thus is slower than the same operation in C.
8. ✅ A BINARY file is generally more compact than an ASCII one.

Ex 11 (1.0 points)

Concerning inter-process communication, indicate which of the following statements is correct. Note that more than one response can be correct and that incorrect answers may imply a penalty on the final score.

Choose one or more options:

1. ⬜ FIFO can only be used between processes sharing an ancestor.
2. ✅ Function `ftok` shares a key among different processes.
3. ✅ A message queue is a linked list of messages stored within the kernel.
4. ⬜ FIFO allows only blocking operations.
5. ⬜ A FIFO differs from a pipe because it includes a linked list of messages stored within the kernel.
6. ✅ Pipes can be used only between processes sharing an ancestor.
7. ⬜ Function `msgctl` sends a message in a message queue.

8. ✅ FIFOs are used to pass streams of anonymous bytes.

Ex 12 (1.0 points)

Analyze the following code snippet. Indicate which of the following statements are correct. Note that more than one response can be correct and that incorrect answers may imply a penalty on the final score.

```
pthread_mutex_lock (&mutex);
count++;
if (count == N_THREAD) {
   sem_post (&sem);
}
pthread_mutex_unlock (&mutex);
sem_wait (&sem);
sem_post (&sem);
```

Scegli una o più alternative:

1. ✅ The mutex named mutex can be substituted by a second semaphore sem2 initialized to 1.
2. ⬜ The piece of code can be used exactly as it is (as a unique synchronization strategy) by a thread that loops through a cycle and hits the barrier at every iteration.
3. ✅ The piece of code includes a barrier implemented in a correct way.
4. ⬜ The variable count must be initialized to 0 by each thread.
5. ⬜ The semaphore `sem` must be initialized to 1.
6. ⬜ The code includes a barrier implemented incorrectly, as it is necessary to insert a cycle to free all waiting threads.
7. ⬜ If the entire construct is used twice, the value of N_THREAD cannot be changed.