Operating Systems – Exercise 2

Processes, Multiprocessing & IPC

Submission & General Guidelines

- Submission deadline is **21/5/2023, 23:55** Moodle server time.
- Submit your answers in the course website only as single ex2-YOUR_ID.**zip (**e.g. ex2-012345678.zip), containing **only**:
    - Ex2_part1q1.**c**
    - Ex2_part1q2.**c**
    - Ex2_part1q3.**c**
    - ~~Ex2_part2q1.**c**~~
    - Ex2_part1q2.**c**
    - part3.**pdf**
- Place your name and ID at the top of every source file, as well as in the PDF with the answers.
- No late submission will be accepted!
- Please give concise answers, but make sure to explain all answers.
- Write **clean code** (readable, documented, consistent, ...).
- Don't forget – According to the course regulations, **your allowed to submit an "I don't know" PDF** file. Doing so will automatically grant you a grade of 50 in the assignment.

Part 1 - Process (32 points)

In this question we will implement different C applications using what we learned regarding multiprocessing. You were given a single template called "2023_ex2_p1.c". You should be able to copy-paste it as is to your C coding environment as instructed in Exercise 1.
Run the code. As you can see, the best known Father - Son dialog in the history of cinema is not printed correctly.

Question 1:
Ok. Let's have an easy start.
Please read about the command "*sleep()*" in the C language.
Use it to change the template code you were given in "2023_ex2.c" so the Epic dialog is printed correctly.

Don't remember it? Or even worse, never seen it ?!?! quickly fix the problem Star Wars: The Empire Strikes Back | "I Am Your Father" | 4K HDR

Save your code in a file called "Ex2_part1q1" and submit it.

Question 2:
Ok, that was easy.
In this question you need to use Inter process communication (IPC) as learned in Tutorial 4.
Please read about the "pipe()" command in the C language.

Use it to get the dialog printed correctly.
You're not allowed to use “sleep()”.
Hint: you may need more than one pipe.
Save your code in a file called “Ex2_part1q2” and submit it.

Question 3:
Edit the following:
Project->Properties->C/C++ Build->Settings->Tool Settings->Cross GCC Linker->Libraries->Add…->”rt”
(do the same with “pthread“ instead of “rt”)

Again!
Same trick. Only this time using semaphores.
The functions you're going to need are: sem_open, sem_wait and sem_post.
Save your code in a file called “Ex2_part1q3” and submit it.

Part 2 - Threads (32 points)

In this question we will implement different C applications using what we learned regarding multiprocessing. You were given a single template called “**2023_ex2_p2.c**”. You should be able to copy-paste it as is to your C coding environment as instructed in Exercise 1.

Make the necessary edits for part 1 question 3 are configured. They are needed here as well

In this part, you're not allowed to change the function “main()” as it’s given to you in the template.

---------- question 1 of part 2 is cancelled and will not be checked ----------
Question 1:

Question 2:
Without changing the function “main”, can you get the dialog to be printed in the correct order?
You may change the “*threadFunc()” as you like.
you're **allowed** to use the “flag” variable.
You’re **Not allowed** to change the printing order withing “threadFunc”.
Use locks.
Save your code in a file called “Ex2_part2q2” and submit it.

Part 3 (36 points) – Theoretical Questions

Question 1

The following code is run by 3 threads:

```
static int x = 0;
void *thread_func(void *p)
{
	x++;
}
```

What is the minimal value of x at the end of the run?

Question 2

Consider the following code:

```
int main ()
{
	int x = 5;
	if ( fork () > 0) {
		x ++;
		fork ();
	} else {
		x - -;
		fork ();
	}
	x - -;
	fork ();
	printf ( %d\ n , x );
	return 0;
}
```

What will be the output if we run the given code? (the order does not matter)

Question 3:

When executing the following code, how many times will the line “forked” appear in the output?

```c
int main()
{
   pid_t wpid;
   int status = 0;
   for ( int i=0; i<10; i++ ){
       if( i % 3 == 0){
           fork();
           }
   }
printf("forked\n");
// wait for all processes
while ((wpid = wait(&status)) > 0);
return 0;
}
```