**CS2106 Operating Systems**

**Assignment 2 – Processes and Threads**

**Answer Book**

|  |  |
| --- | --- |
| Member 1 Name: Wang Riwu | Member 1 Matric No: A0135766W |
| Member 2 Name: Jeremy Heng Wen Ming | Member 2 Matric No: A0146789H |

Question 1

This is what I see on the screen: Parent sent message: Hello child! and 128

My single statement description is: Creates a child process and send 2 values to the child using pipes for the child to print it to stdout.

Question 2

The sizeof function returns the size of a variable or datatype.

Question 3

My completed code is attached below: //add after code finalised

Question 4

The threads print out of order. The reason is threads may get pre-empted by other threads before they finish executing.

Question 5

The threads do share memory. Referring to ctr, I conclude this because value of ctr can increment to up to 9, which would not be possible if the threads each have their own portions of memory. Hence they must be sharing memory.

Question 6

The values of ctr as printed by the threads are wrong. The reason is that the threads might be pre-empted by other threads before the increment of ctr is executed. Furthermore the value ctr could be subjected to race condition, resulting in some threads not correctly incrementing the value.

Question 7

The variable "i" must be cast into void \* because that’s the data type of the parameter of pthread\_create, hence it must be explicitly cast to the data type that matches the parameter type of the function.

In child it does not have to be cast back into int because it is implicitly converted to int in printf using %d.

Question 8

The changes I made are calling pthread\_join after pthread\_create to wait for it to finish executing.

My code is attached here: //add after code finalised

Question 9

The value of glob printed by main is 20.

Question 10

The changes we made are

1. declare a variable pthread\_t thread[10]
2. change child((void \*) i); to pthread\_create(&thread[i], NULL, child, (void \*) i);

Question 11

The value printed is incorrect. This is because the printf statement in main sometimes executes before all the threads finish executing. Also the variable glob can be subjected to race condition as the threads might attempt to modify it concurrently.

Question 12

The threads now update glob correctly. This is because before the threads goes into modifying the glob variable, they are blocked until they successfully obtains the mutex lock, at which point they will prevent other threads from successfully obtaining the same mutex lock until they finish modifying the glob variable and frees the mutex lock, hence preventing race conditions.

However the value of glob printed by main is still incorrect as printf in main might still execute before all the threads finish executing.

Question 13

The changes we made were adding

for(i=0; i<10; i++)

pthread\_join(thread[i], NULL);

before the printing of glob in main to wait for all the threads to finish executing.

Our program is attached below: //add after code finalised