Set_G_Lab5

Page 1:

1 ✓ 2 ✓ 3 ✓
4 ✓ 5 ✓ 6 ✓
7 ✓ 8 ✓ 9 ✓
10 ✓ 11 ✓

Quiz Information

Question 1 (1 point) ✓ *Saved*

Use cocalc.com to run the code given below and answer the following questions(Q2, Q3, Q4, Q5, Q6):

```
gcc main1.c -lpthread -o main1.out
./main1.out & pstree -p | grep main1.out
```

main1.c

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
int fd, counter = 0;
void *myMethod(void *arg){
    int * id;
    id = (int *)arg;
    char st[]=" [%c] Hello from thread_%d.\n";
        for(int i=0; i<5; i++ ){
            sprintf(st, st, *id==1? '*': '+', *id);
            write(fd, st, strlen(st));
        }
        printf("Thread %d counter = %d \n", *id, ++counter);
        sleep(1);
    return NULL;
}

int main(int argc, char *argv[]){
    fd = open("myfile.txt",'a');
    if (fd<0){
        printf("If you see this message, delete myfile.txt using rm -f myfile.txt, then run the program again.\n");
        exit(0);
    }
    pthread_t threadMyMethodOne;
    pthread_t threadMyMethodTwo;
    int first_thread = 1, second_thread = 2;
    pthread_create(&threadMyMethodOne, NULL, myMethod, &first_thread);
    pthread_create(&threadMyMethodTwo, NULL, myMethod, &second_thread);
    pthread_join(threadMyMethodOne, NULL);
    pthread_join(threadMyMethodTwo, NULL);
    printf("The file (fd=%d) is  closed.\n", fd);
    close(fd);
    return 0;
}
```

Insert a screenshot of your output.

Paragraph

```
~$ gcc main1.c -lpthread -o main1.out
~$ ./main1.out & pstree -p | grep main1.out
[1] 454
Thread 1 counter = 1
Thread 2 counter = 2
                          |           |-main1.out(454)-+-{main1.out}(457)
                          |           |                `-{main1.out}(458)
~$ The file (fd=3) is closed.
```

Set_G_Lab5

Page 1:

1 2 3
4 5 6
7 8 9
10 11

Quiz Information

Add a File | Record Audio | Record Video

Question 2 (1 point) ✓ *Saved*

After running the program, insert a screenshot showing the contents of myfile.txt. You may need to change the file's permissions using a command like **chmod 444 myfile.txt**.

Paragraph

myfile.txt

```
1  [*] Hello from thread_1.
2  [*] Hello from thread_1.
3  [*] Hello from thread_1.
4  [*] Hello from thread_1.
5  [*] Hello from thread_1.
6  [+] Hello from thread_2.
7  [+] Hello from thread_2.
8  [+] Hello from thread_2.
9  [+] Hello from thread_2.
10 [+] Hello from thread_2.
11
```

Add a File | Record Audio | Record Video

Question 3 (2 points) ✓ *Saved*

Based on the output of your code, do you think that the global variable "counter" is shared between threads 1 and 2 (Y/N)?

Y

From the contents of myfile.txt, is the file myfile.txt shared between the two threads (Y/N)? Y

Set_G_Lab5

Page 1:

1 2 3
4 5 6
7 8 9
10 11

Quiz Information

Question 4 (1 point) Saved

Change the sleep time from 1 to 100, then recompile your program and run it using

```
./main1.out & pstree -p | grep main1.out
```

From the output, Obtain the *pid* and *tid* for the process and threads. Insert a screenshot of your output.

Paragraph

```
$ ./main1.out & pstree -p | grep main1.out
[1] 788
Thread 1 counter = 1
Thread 2 counter = 2
                          |-main1.out(788)-+-{main1.out}(791)
                          |                `-{main1.out}(792)
$ The file (fd=3) is closed
```

pid: 788

tid: 791, 792

Add a File Record Audio Record Video

Question 5 (1 point) Saved

After running (make sure the sleep time is already changed from 1 to 100)

```
./main1.out & pstree -p | grep main1.out
```

In the second terminal use **kill -9 <tid>** to terminate one of the threads. Does the main process remain alive after the thread is killed (Y/N)? Y

Question 6 (1 point) Saved

If a process creates two threads, which of the following statements is true?

- ○ Each thread receives a completely new PID and no TID
- ◉ Both threads share the same PID but have different TIDs
- ○ Both threads have different PIDs but identical TIDs

Q7

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <ucontext.h>
#include <sys/syscall.h>

#define STACK_SIZE 8192
ucontext_t main_ctx, th1_ctx, th2_ctx;


void thread_1() {
```

```
    int c = 0;
    while(1){
        c++;
        printf("Thread 1.\n");
        if (c%5==0){
            sleep(1);
            swapcontext(&th1_ctx, &th2_ctx);
        }
    }
}

void thread_2() {
    int c = 0;
    while(1){
        c++;
        printf("Thread 2.\n");
        if (c%3==0){
            sleep(1);
            swapcontext(&th2_ctx, &th1_ctx);
        }
    }
}

int main() {
    // Allocate stacks
    char *sum_stack = malloc(STACK_SIZE);
    char *sort_stack = malloc(STACK_SIZE);

    if (!sum_stack || !sort_stack) {
        perror("malloc");
        exit(1);
    }

    // Create sum thread context
    getcontext(&th1_ctx);
    th1_ctx.uc_link = 0;
    th1_ctx.uc_stack.ss_sp = sum_stack;
```

```
    th1_ctx.uc_stack.ss_size = STACK_SIZE;
    makecontext(&th1_ctx, thread_1, 0);

    // Create sort thread context
    getcontext(&th2_ctx);
    th2_ctx.uc_link = 0;
    th2_ctx.uc_stack.ss_sp = sort_stack;
    th2_ctx.uc_stack.ss_size = STACK_SIZE;
    makecontext(&th2_ctx, thread_2, 0);

    setcontext(&th1_ctx);

    free(sum_stack);
    free(sort_stack);
    return 0;
}
```

Set_G_Lab5

Page 1:

1 2 3
4 5 6
7 8 9
10 11

Quiz Information

Insert a screenshot of your output.

Paragraph

```
Thread 1.
Thread 1.
Thread 1.
Thread 1.
Thread 1.
Thread 2.
Thread 2.
Thread 2.
Thread 1
```

Add a File | Record Audio | Record Video

Question 8 (2 points) ✓ *Saved*

Which thread starts first? e.g. 1 or 2 [1]

Modify thread_1() and thread_2() to print the thread ID using

```
printf("Thread 1.\n"); printf("Thread 1. PID: %d TID: %ld \n", getpid(), syscall(SYS_gettid));
printf("Thread 2.\n"); printf("Thread 2. PID: %d TID: %ld \n", getpid(), syscall(SYS_gettid));
```

Does each thread have a different thread ID (Y/N)? [N]

Change the sleep(1) calls to sleep(100) in both thread functions.

Does one sleep call block both threads (Y/N)? [Y]

Does one sleep call block the main process (Y/N)? [Y]

Question 9 (1 point) ✓ *Saved*

A process uses two user-level threads managed entirely in user space. Which of the following statements is true regarding the PID and TIDs seen by the Linux kernel?

- ○ The kernel assigns a different PID to each user-level thread
- ○ Each user-level thread has a unique TID, but they all share the same PID
- ◉ The process has one PID, and both user-level threads share the same TID as the PID

Question 10 (1 point) ✓ *Saved*

Use the online C compiler

https://www.onlinegdb.com/online_c_compiler

to run your code and then answer the following questions.

```c

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

int global_var = 0;

int main(){
  int status;
  int pid_t, fork_return;
  fork_return = fork();

  if (fork_return==0){
    global_var++;
    printf("Child  pid  = %d \t global_var = %d\n", getpid(), global_
  }else{
    global_var++;
    waitpid(-1, &status, 0);
    printf("Parent pid  = %d \t global_var = %d\n", getpid(), global_
  }
  return 0;
}
```

Insert a screenshot of your output.

Set_G_Lab5

Page 1:

1 2 3
4 5 6
7 8 9
10 11

Quiz Information

```
23    return 0;
24 }
```

Insert a screenshot of your output.

Paragraph

```
Child  pid  = 1237       global_var = 1
Parent pid  = 1233       global_var = 1
```

Add a File | Record Audio | Record Video

Question 11 (1.5 points) ✓ *Saved*

From the output, determine the value of the global_var variable in the parent process. e.g. 2

1

From the output, determine the value of the global_var variable in the child process. e.g. 2

1

Based on your observations, do you think that the global_var variable is shared between the parent and child processes (Y/N)?

N

Submit Quiz *11 of 11 questions saved*