Sistemas Operativos

Concurso 3 – Sinais/Threads/Mutex

67990 – José Diogo Ferrás

Ex. I

a)

Código:

```c
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>

int i = 0;

void sighandler() {
    i++;
    if(i < 5){
        printf("\nnice try\n");
    }
    else{
        printf("\nyou got me this time\n");
    }
}

int main (int argc, char* argv[]){
    while(1){
        if (signal(SIGINT,&sighandler) == SIG_ERR ){
            printf ( "NO  signal\n" ) ;
        }
        if(i == 5){
            break;
        }
    }
    return 0 ;
}
```

I/O:

```
zediogo@ubuntu:~/Documents/c3$ ./a.out
^C
nice try
^C
nice try
^C
nice try
^C
nice try
^C
you got me this time
zediogo@ubuntu:~/Documents/c3$
```

b)

Código:

```c
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

int i = 0;
char msg[] = "mensagem da thread ";

void *routine(){
    for(int j = 0; j < 19; j++){
        printf("%c", msg[j]);
        fflush(stdout);
        sleep(1);
    }
    printf("%d\n", i);
    sleep(1);
}

int main (int argc, char* argv[]){
    pthread_t th[5];
    for(i = 0; i < 5; i++){
        if(pthread_create(th + i, NULL, &routine, NULL)){
            return 1;
        }
        if (pthread_join(th[i], NULL)){
            return 2;
        }
    }
    return 0 ;
}
```

I/O:

```
zediogo@ubuntu:~/Documents/c3$ ./a.out
mensagem da thread 0
mensagem da thread 1
mensagem da thread 2
mensagem da thread 3
mensagem da thread 4
```

Ex. II

Código:

```c
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <stdbool.h>

bool end = false;
char msg[30];

void *routine0(){
    printf("Thread 1: ");
    fflush(stdout);
    fgets(msg, sizeof(msg), stdin);
}

void *routine1(){
    if(msg[0] == 'E' && msg[1] == '\n'){
        end = true;
    }
    for(int i = 0; i < 30; i++){
        if(msg[i] == '\n'){
            break;
        }
        printf("Thread 2: ");
        printf("%c\n", msg[i]);
    }
}
```

```c
int main (int argc, char* argv[]){
    pthread_t th0, th1;
    while(true){
        if(pthread_create(&th0, NULL, &routine0, NULL)){
            return 1;
        }
        if (pthread_join(th0, NULL)){
            return 2;
        }
        if(pthread_create(&th1, NULL, &routine1, NULL)){
            return 1;
        }
        if (pthread_join(th1, NULL)){
            return 2;
        }
        if(end){
            break;
        }
    }
    printf("Prog finished\n");
    return 0;
}
```

I/O:

```
zediogo@ubuntu:~/Documents/c3$ ./a.out
Thread 1: abc
Thread 2: a
Thread 2: b
Thread 2: c
Thread 1: giro
Thread 2: g
Thread 2: i
Thread 2: r
Thread 2: o
Thread 1: EEE
Thread 2: E
Thread 2: E
Thread 2: E
Thread 1: E
Thread 2: E
Prog finished
```

Ex. III

a)

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

pthread_t tid[2];
int counter;

void * trythis(void* arg){
    unsigned long i = 0;
    counter += 1;
    printf("\nJob %d has started\n", counter);
    for(i = 0; i < (0xFFFFFFFF); i++);
    printf("\n Job %d has finished\n", counter);
    return NULL;
}

int main(void){
    int i = 0;
    int error;
    while(i < 2){
        error = pthread_create(&(tid[i]), NULL, &trythis, NULL);
        if(error != 0){
            printf("\nThread can't be created:[%s]", strerror(error));
        }
        i++;
    }
    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);
    return 0;
}
```

```
zediogo@ubuntu:~/Documents/c3$ ./a.out

Job 1 has started

Job 2 has started

 Job 2 has finished

 Job 2 has finished
```

b)

Isto acontece pois o counter é incrementado pela primeira vez quando a primeira thread é criada, e é incrementado pela segunda vez quando a segunda thread é incrementada.

Como podemos observar, a segunda thread começa a executar a sua operação antes da primeira thread acabar e como counter é uma variável global, esta é alterada na thread 1 quando é alterada na thread 2, logo quando acaba de executar a tarefa da thread 1, o valor do counter já está a 2 e printa “Job 2 has finished” também.

c)

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

pthread_t tid[2];
int counter;
pthread_mutex_t mutex;

void * trythis(void* arg){
    unsigned long i = 0;
    pthread_mutex_lock(&mutex);
    counter += 1;
    printf("\nJob %d has started\n", counter);
    for(i = 0; i < (0xFFFFFFFF); i++);
    pthread_mutex_unlock(&mutex);
    printf("\n Job %d has finished\n", counter);
    return NULL;
}

int main(void){
    pthread_mutex_init(&mutex, NULL);
    int i = 0;
    int error;
    while(i < 2){
        error = pthread_create(&(tid[i]), NULL, &trythis, NULL);
        if(error != 0){
            printf("\nThread can't be created:[%s]", strerror(error));
        }
        i++;
    }
    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);
    pthread_mutex_destroy(&mutex);
    return 0;
}
```