Bahria University



Lahore Campus

# Final Term Exams (spring -2024 Semester)

*Department of Computer Sciences*

|  |
| --- |
| Paper Show Date & Time: |

|  |  |  |  |
| --- | --- | --- | --- |
| **Exam Date & Time** | **13-06-2024/8:300 AM** | **Session** | **i** |
| **Instructor Name** | **Abdullah** | **Program/Semester:** | **BSCS/5A** |
| **Course Title** | **Operating System Lab** | **Course Code:** | **CSL-320** |
| **Time Allowed** | **90 mins** | **Max Marks** | **30** |

**Instructions:**

**Read out the Instructions carefully.**

1. **Read the exam carefully and attempt all the questions**
2. **Understanding of all questions is the part of the exam**
3. **Copied answers will straight away be awarded with *ZERO***
4. **For Output you need to take full screen *Screenshot* and paste it in the given space.**
5. **Submission method is as follows; it should be strictly followed otherwise marks will be deducted**

**Make a Folder to add this word solution file in the folder naming “*Name\_Enroll\_Midterm\_Section*” along with the .c and. s files of each programming question. The .c and. s file name should be saved according to the question no i.e. “*Q1.s*”.**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Student Name** …AFFAN AHMAD…. **Enrollment Number**:…03-134221-003…

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation of CLO** | **Ques # / Part #** | **Marks** | **Obtained Marks** |
| **CLO1: Implement OS concepts like those of shell scripting, processes, file-manipulation and inter-process communication** | 1 | 10 |  |
| 2 | 10 |
| 3 | 10 |
| **CLO2: Demonstrate the knowledge in applying system software and tools available in modern  operating systems** | 1,2,3 | 10 |  |
|  | |  |  |

**Question No 01:**  Write a C program that uses the sigaction() function to handle the SIGINT and SIGQUIT signals. The program should print a message when the user presses Ctrl+C and then ignore the next Ctrl+C press. The program should also print a message when the user presses Ctrl+\ and then exit gracefully. **[10 marks]**

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

#include <unistd.h>

int count = 0;

void handle\_sigint(int sig) {

count++;

printf("You pressed Ctrl-C %d time(s)\n", count);

if (count == 2) {

signal(SIGINT, SIG\_IGN);

}

}

void handle\_sigquit(int sig) {

printf("You pressed Ctrl-\\, exiting the program\n");

exit(0);

}

int main() {

struct sigaction sa\_int;

sa\_int.sa\_handler = handle\_sigint;

sa\_int.sa\_flags = 0;

sigaction(SIGINT, &sa\_int, NULL);

struct sigaction sa\_quit;

sa\_quit.sa\_handler = handle\_sigquit;

sa\_quit.sa\_flags = 0;

sigaction(SIGQUIT, &sa\_quit, NULL);

while (1) {

printf("Hello, world!\n");

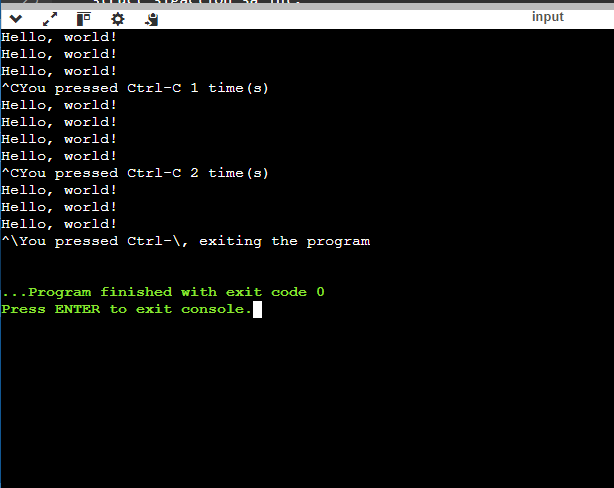
sleep(1);

}

return 0;

}

**OUTPUT:**



**IN C++:**

#include<iostream>

using namespace std;

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

#include <unistd.h>

int count = 0;

void handle\_sigint(int sig) {

count++;

cout <<"You pressed Ctrl-C time(s)"<< count<< endl;

if (count == 2) {

signal(SIGINT, SIG\_IGN);

}

}

void handle\_sigquit(int sig) {

cout << "You pressed Ctrl, exiting the program"<< endl;

exit(0);

}

int main() {

struct sigaction sa\_int;

sa\_int.sa\_handler = handle\_sigint;

sa\_int.sa\_flags = 0;

sigaction(SIGINT, &sa\_int, NULL);

struct sigaction sa\_quit;

sa\_quit.sa\_handler = handle\_sigquit;

sa\_quit.sa\_flags = 0;

sigaction(SIGQUIT, &sa\_quit, NULL);

while (1) {

cout << "Hello, world!\n";

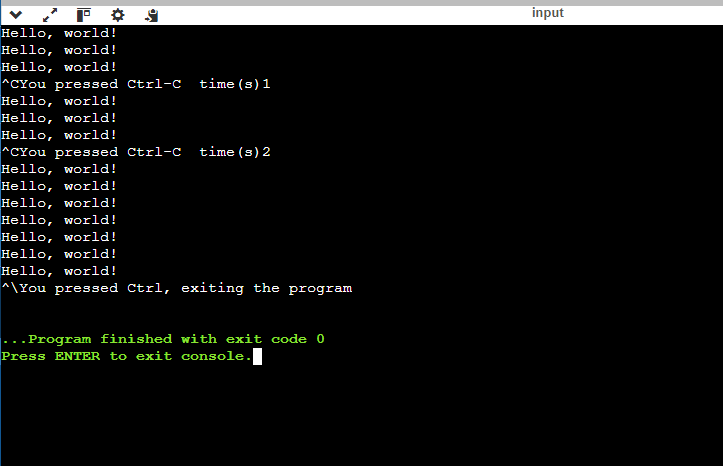
sleep(1);

}

return 0;

}

**Output:**



**Question No 02:**  Write a C program that uses two semaphores and a mutex to synchronize three threads: A, B, and C. The threads should print their names in the following order: A, B, C, A, B, C, … The program should run indefinitely until the user interrupts it. **[10 marks]**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#define N 3

char \*names[N] = {"A", "B", "C"};

sem\_t sems[N];

pthread\_mutex\_t mutex;

void \*thread\_func(void \*arg) {

int id = \*((int \*) arg);

while (1) {

sem\_wait(&sems[id]);

pthread\_mutex\_lock(&mutex);

printf("%s\n", names[id]);

pthread\_mutex\_unlock(&mutex);

sem\_post(&sems[(id + 1) % N]);

}

}

int main() {

int ids[N];

pthread\_t threads[N];

pthread\_mutex\_init(&mutex, NULL);

for (int i = 0; i < N; i++) {

sem\_init(&sems[i], 0, i == 0 ? 1 : 0);

}

for (int i = 0; i < N; i++) {

ids[i] = i;

pthread\_create(&threads[i], NULL, thread\_func, &ids[i]);

}

for (int i = 0; i < N; i++) {

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

}

pthread\_mutex\_destroy(&mutex);

for (int i = 0; i < N; i++) {

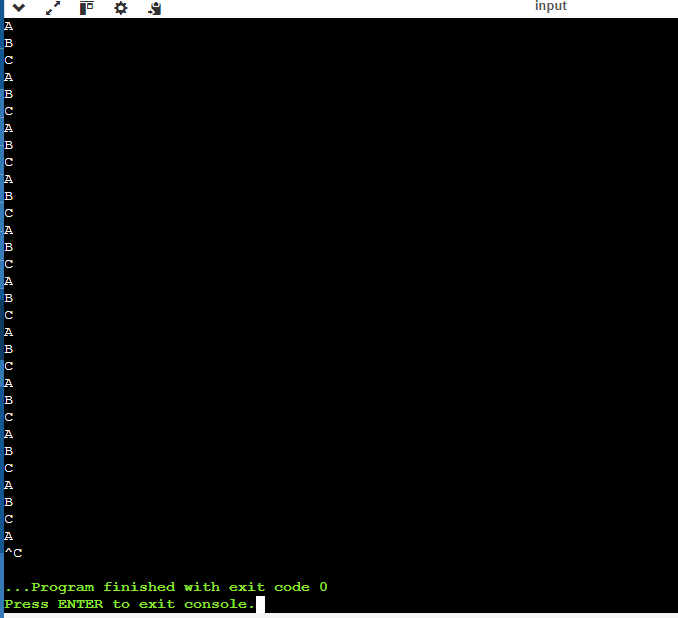
sem\_destroy(&sems[i]);

}

return 0;

}

OUTPUT:



**IN C++:**

#include<iostream>

using namespace std;

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#define N 3

char \*names[N] = {"A", "B", "C"};

sem\_t sems[N];

pthread\_mutex\_t mutex;

void \*thread\_func(void \*arg) {

int id = \*((int \*) arg);

while (1) {

sem\_wait(&sems[id]);

pthread\_mutex\_lock(&mutex);

cout << names[id]<< endl;

pthread\_mutex\_unlock(&mutex);

sem\_post(&sems[(id + 1) % N]);

}

}

int main() {

int ids[N];

pthread\_t threads[N];

pthread\_mutex\_init(&mutex, NULL);

for (int i = 0; i < N; i++) {

sem\_init(&sems[i], 0, i == 0 ? 1 : 0);

}

for (int i = 0; i < N; i++) {

ids[i] = i;

pthread\_create(&threads[i], NULL, thread\_func, &ids[i]);

}

for (int i = 0; i < N; i++) {

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

}

pthread\_mutex\_destroy(&mutex);

for (int i = 0; i < N; i++) {

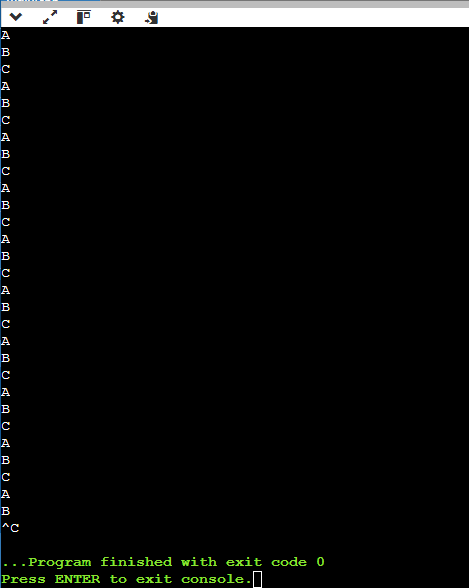
sem\_destroy(&sems[i]);

}

return 0;

}

Output:



Code Exit With Ctrl+P

:

#include<iostream>

#include <csignal> // For signal handling

using namespace std;

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#define N 3

char \*names[N] = {"A", "B", "C"};

sem\_t sems[N];

pthread\_mutex\_t mutex;

volatile sig\_atomic\_t stop\_flag = 0; // Flag to indicate if Ctrl+P is pressed

void signal\_handler(int signum) {

if (signum == SIGINT) {

stop\_flag = 1;

cout << "Ctrl+P pressed. Exiting gracefully..." << endl;

}

}

void \*thread\_func(void \*arg) {

int id = \*((int \*) arg);

while (!stop\_flag) {

sem\_wait(&sems[id]);

pthread\_mutex\_lock(&mutex);

cout << names[id]<< endl;

pthread\_mutex\_unlock(&mutex);

sem\_post(&sems[(id + 1) % N]);

}

pthread\_exit(NULL);

}

int main() {

int ids[N];

pthread\_t threads[N];

pthread\_mutex\_init(&mutex, NULL);

for (int i = 0; i < N; i++) {

sem\_init(&sems[i], 0, i == 0 ? 1 : 0);

}

signal(SIGINT, signal\_handler);

for (int i = 0; i < N; i++) {

ids[i] = i;

pthread\_create(&threads[i], NULL, thread\_func, &ids[i]);

}

for (int i = 0; i < N; i++) {

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

}

pthread\_mutex\_destroy(&mutex);

for (int i = 0; i < N; i++) {

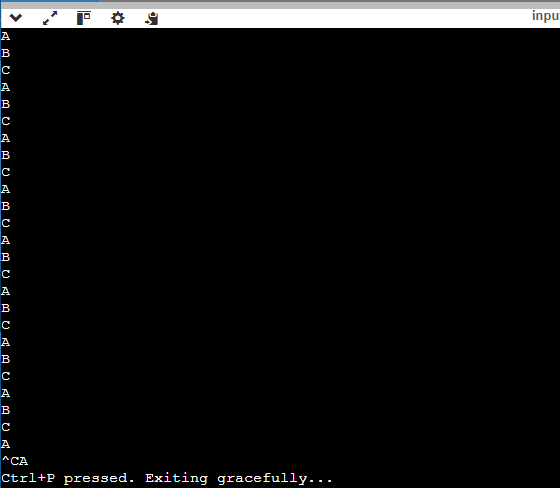
sem\_destroy(&sems[i]);

}

return 0;

}

OUTPUT:



**Question No 03:**  Write a C/C++ program for synchronization of the threads in Task 1. Use Mutex lock and unlock. Make sure the Thread 1 completes its computation than Thread 2 starts its execution. **[10 Marks]**

**#include<iostream>**

**using namespace std;**

**#include <pthread.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <unistd.h>**

**pthread\_t tid[2];**

**int counter;**

**void\* trythis(void\* arg)**

**{**

**unsigned long i = 0;**

**counter += 1;**

**cout << " Job "<< counter<<" has started"<< endl;**

**for (i = 0; i < (0xFFFFFFFF); i++)**

**;**

**cout <<" Job "<< counter << " has finished"<< endl;**

**return NULL;**

**}**

**int main(void)**

**{**

**int i = 0;**

**int error;**

**while (i < 2) {**

**error = pthread\_create(&(tid[i]), NULL, &trythis, NULL);**

**if (error != 0)**

**cout <<"Thread can't be created : "<< strerror(error)<< endl;**

**i++;**

**}**

**pthread\_join(tid[0], NULL);**

**pthread\_join(tid[1], NULL);**

**return 0;**

**}**

**Output:**

