CS 259 - HW 2

Name : Hitesh Kumar

SJSU ID : 015237989

Solution 1

For N = 1,2,3,4,5, the system is deadlock free as

1. each process will use their exclusive tape drives without any dependency on the other tape drives.
2. Hence, no condition for deadlock will arise.

Solution 2(a)

There is no cycle formed in any direction, there is no deadlock.

There is no circular wait condition formed, hence there is no deadlock.

Solution 2(b)

There is no cycle formed in any direction, there is no deadlock.

There is no circular wait condition formed, hence there is no deadlock.

Solution 3(a)

Horizon bounds represents is Process A's execution .

Vertical bounds represents is Process B's execution

When the trajectory is at point 't' and it execute process A, then it requests for plotter which is being allocated to process B, if it goes in vertical section, it requests Printer which is assigned to Process A, hence it should not enter the unsafe region as it will go to deadlock. **Hence if its reaches to intersection of I2 and I6, it will go to deadlock.**

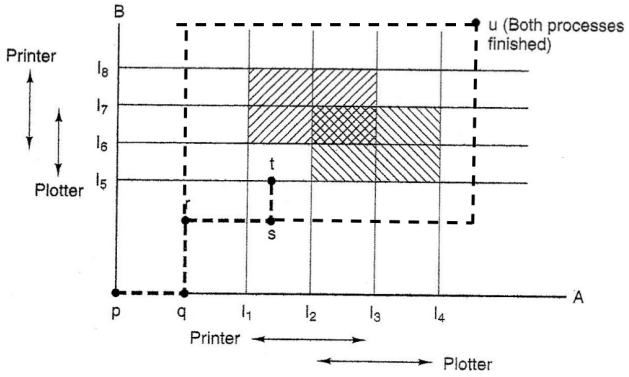
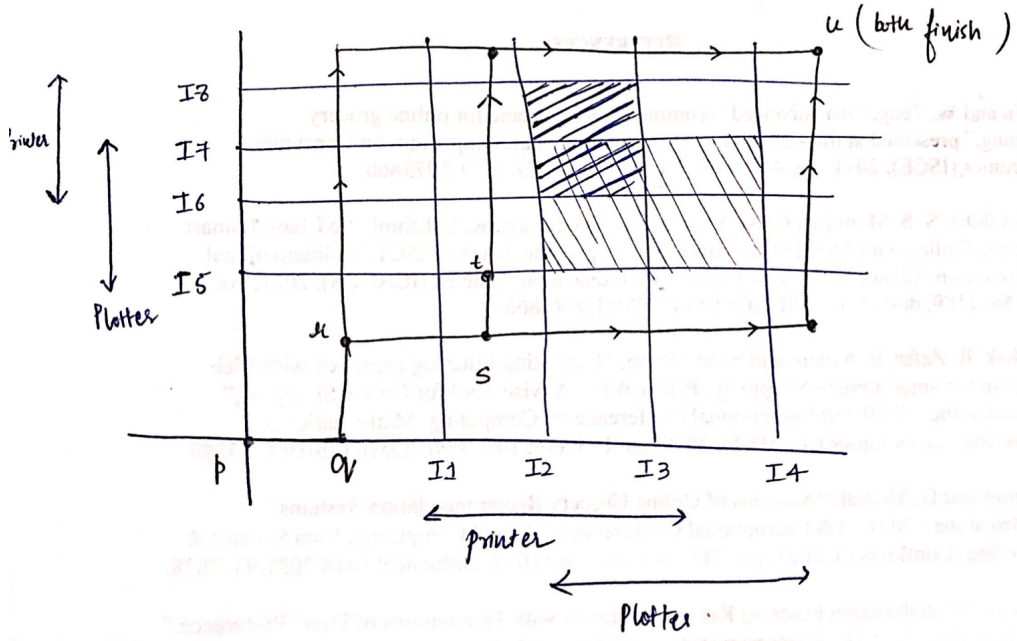


Figure 3-12. Two process resource trajectories.

Solution 3(b) :

There is no way the trajectory can reach intersection of I3 and I7, because if the trajectory enters the unsafe region, it will go into the deadlock state and the it will never able to each I3 and I7.

Solution 3(c) :



Solution 3(d) :

If the trajectories are diagonal , it's a multicore processor hardware platform.

Solution 4

No flowchart doesn't provide any mutual exclusion.

In flowchart (a), deadlock will take place due to the following sequence occurences:

(i) Process 1 executes till P1 value is True

OS shifts to Process 2

(ii) Process 2 executes till P2 value is True

(iii) Process 2 will wait on executing the condition since P1 value is True.

OS halts Process 2 and swaps to Process 1.

(iv) Process 1 also waits on executing the condition since P2 is True.

Therefore, a deadlock occurs and no process is able to use the shared resource X.

In flowchart (b), mutual exclusion condition is violated.

Below sequence are happened :

(i) Process 1 executes till the condition statement

(ii) Process 2 executes the condition statement; until P1 =FALSE

(iii) Process 2 executes P2 = True , Read X and Update X

(iv) Process 1 executes P1= True and READ X and UPDATE X

Solution 5

No, this protocol will be not be effective in circumventing deadlock while effectively utilizing resources.

Scenario 1:

If both the person arrive at the same time 't' , both the side keeps waiting for the other side to go first as per the condition. This also creates the deadlock in the given system

Scenario 2:

If both the person might want to cross the river at the same time, then both person will meet in the middle.

Solution 6

1) Mutual Exclusion

Only a single process at a time can use the resource. If process-2 requests the resource, process-2 has to wait until the resource is released.

2) Hold and Wait

Processes holding resources granted earlier can request new resources. If process must wait for a new additional resource, it continues to hold onto the resources it currently has.

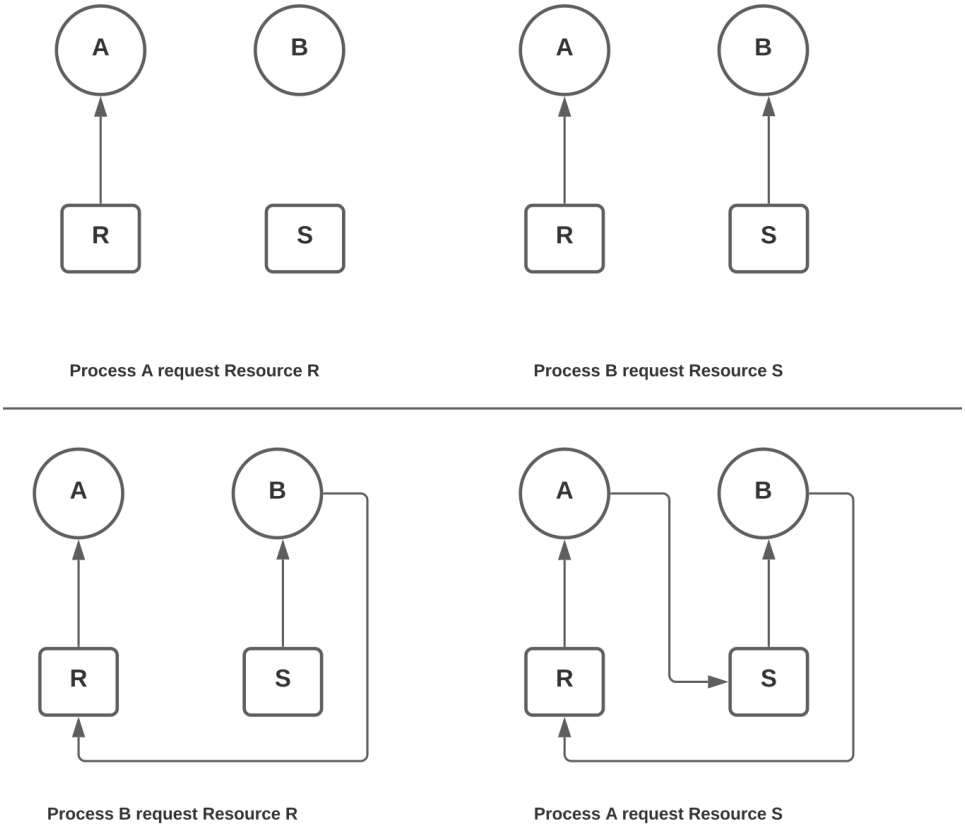
3) No Preemption

Resources previously granted to a process cannot be forcibly taken away from that process.
Resources can only be released voluntarily by the processes that are holding them.

4) Circular Wait.

There must be a circular chain of two or more processes. Each is waiting for a resource held by the next member of the chain.

Solution 7



Yes, the system is in deadlock since there is a closed loop that got created in the resource allocation graph. Each process is waiting for the other process to release the resource. Hence, it's a deadlock.

Solution 8(a)

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>
int main(){
for (int i = 0 ; i<8;i++)
{
printf("Sleep #%d\n",i+1);
sleep(1);
}
```

```
printf("Program exiting\n");
}
```

Console output :

```
(base) hiteshkumar@Hiteshs-MacBook-Pro C work % ./assign2_8a
Sleep #1
Sleep #2
Sleep #3
Sleep #4
^C
(base) hiteshkumar@Hiteshs-MacBook-Pro C work %
```

Explanation : As soon as Control+C is pressed, the execution is stopped. The program is terminated.

Solution 8(b)

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>
#include<signal.h>
int main ()
{
signal(SIGINT, SIG_IGN);
for (int i = 0 ; i<10;i++)
{
printf("Sleep #%d\n",i+1);
sleep(1);
}
printf("Program exiting\n");
}
```

Console output :

```
(base) hiteshkumar@Hiteshs-MacBook-Pro C work % ./assign2_8b
Sleep #1
Sleep #2
Sleep #3
^CSleep #4
Sleep #5
^CSleep #6
Sleep #7
^CSleep #8
Sleep #9
Sleep #10
Program exiting
(base) hiteshkumar@Hiteshs-MacBook-Pro C work %
```

Explanation : As soon as Control+C is pressed, the execution continues. The program keeps on running and ignores the Control+C pressed any number of times. This is because the signal is ignored using **signal(SIGINT, SIG_IGN);** line statement.

Solution 8(c)

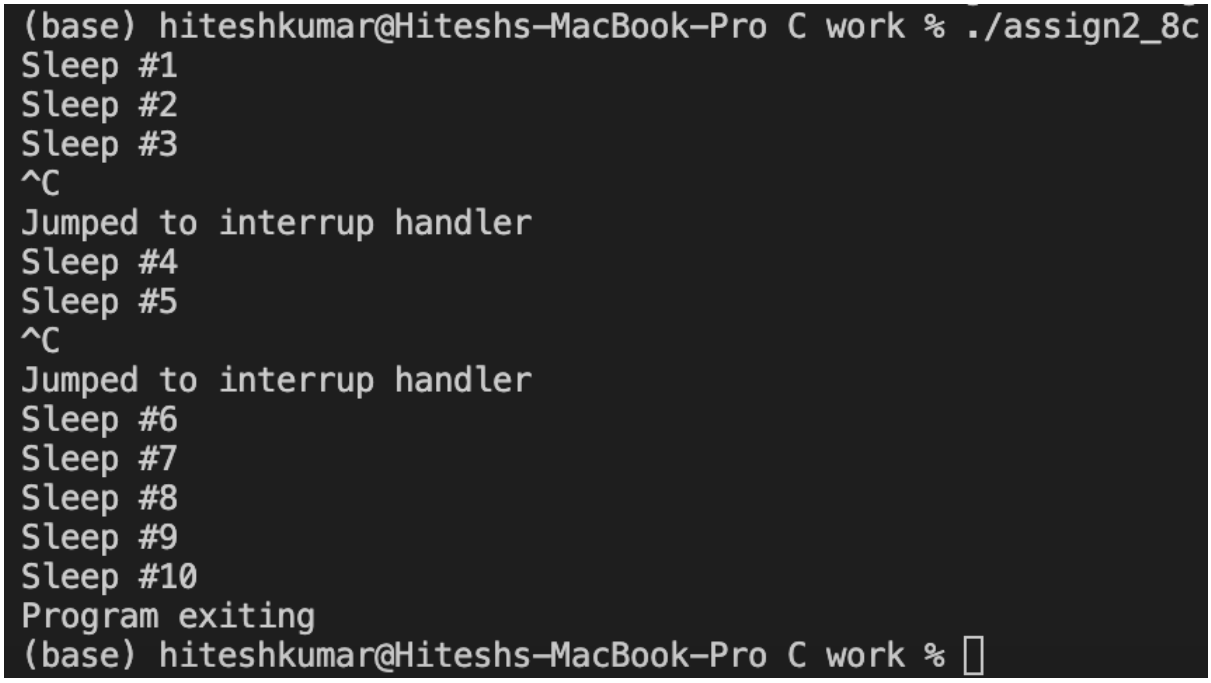
```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>
#include<signal.h>
void sigintHandler(int sig_num)
{
//signal(SIGINT, sigintHandler);
printf("\nJumped to interrup handler\n");
fflush(stdout);
}
int main ()
{
signal(SIGINT, sigintHandler);
```

```
for (int i = 0 ; i<10;i++)
{
printf("Sleep #%d\n",i+1);
sleep(1);
}
printf("Program exiting\n");
}
```

Explanation :

When Control+C is pressed, the execution jumps to interrupt handler and the printf message is printed. The execution continues as per defined .

Output :



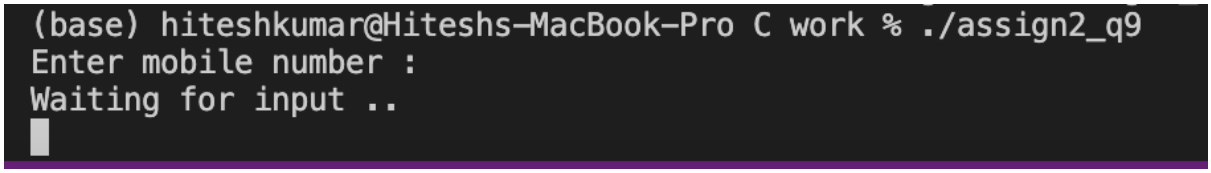
Solution 9

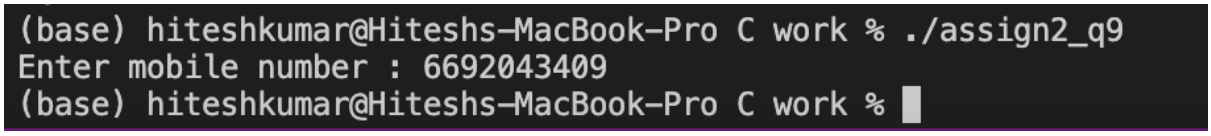
Code :

```
#include<stdio.h>
```

```
#include<unistd.h>
#include<signal.h>
void sig_handler(int signum){
printf("\nWaiting for input ..\n");
}
int main(){
signal(SIGALRM,sig_handler);
alarm(10);  // Scheduled - 10 seconds
int i;
printf("Enter mobile number : ");
scanf("%d",&i);
alarm(0);
return 0;
}
```

Output :





Explanation :

In the above console output, the user needs to input their mobile number and if the user doesn't enter, there is a message "Waiting for input" that is generated on the console.

Once the user inputs, the program is finished completely.

Solution 10(a)

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<unistd.h>
#define buffer_SIZE 16
int main()
{
//pipe descriptors
char mesg[buffer_SIZE];
char buffer[buffer_SIZE];
int x,y,m2,y_m2;
int pipefd[2];
if(pipe(pipefd) == -1){
printf("Error");
}
printf("\nEnter the message #1 :");
scanf("%d",&x);
write(pipefd[1], &x, sizeof(int));
read(pipefd[0], &y, sizeof(int));
printf("message #1 received : %d",y);
//--------------------m2 ----------
printf("\n\nEnter the message #2 :");
scanf("%d",&x);
write(pipefd[1], &x, sizeof(int));
```

```
read(pipefd[0], &y, sizeof(int));
printf("message #2 received : %d",y);
//close(pipefd[1]);
//--------------------m3 --------------
printf("\n\nEnter the message #3 :");
scanf("%d",&x);
write(pipefd[1], &x, sizeof(int));
read(pipefd[0], &y, sizeof(int));
printf("message #3 received : %d",y);
//close(pipefd[1]);
//--------------------m4 --------------
printf("\n\nEnter the message #4 :");
scanf("%d",&x);
write(pipefd[1], &x, sizeof(int));
read(pipefd[0], &y, sizeof(int));
printf("message #4 received : %d",y);
//close(pipefd[1]);
return 0;
}
```

Output :

```
(base) hiteshkumar@Hiteshs-MacBook-Pro C work % ./assign10

Enter the message #1 :10
message #1 received : 10

Enter the message #2 :20
message #2 received : 20

Enter the message #3 :30
message #3 received : 30

Enter the message #4 :40
message #4 received : 40%
```

Explanation :

The four messages are taken one by one using pipex

Solution 10(b)

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#define BUFFERSIZE 20
int main(){
int pipeData[2];
char msg[BUFFERSIZE];
char buf[BUFFERSIZE];
if(pipe(pipeData) == -1){
perror("pipe error occured");
exit(EXIT_FAILURE);
}
//writing starts
if(fork() == 0){
//child
```

```
close(pipeData[1]);
read(pipeData[0], buf, BUFFERSIZE);
printf("This is child process reading first message. Content is :%s\n", buf);
read(pipeData[0], buf, BUFFERSIZE);
printf("This is child process reading second message. Content is :%s\n", buf);
read(pipeData[0], buf, BUFFERSIZE);
printf("This is child process reading third message. Content is :%s\n", buf);
read(pipeData[0], buf, BUFFERSIZE);
printf("This is child process reading fourth message. Content is :%s\n", buf);
close(pipeData[0]);
exit(EXIT_SUCCESS);
}
// parent process
close(pipeData[0]);
sprintf(msg , "San");
printf("This is parent process. Writing first message into pipe\n");
write(pipeData[1], msg, BUFFERSIZE);
sprintf(msg , "Jose");
printf("This is parent process. Writing second message into pipe\n");
write(pipeData[1], msg, BUFFERSIZE);
sprintf(msg , "State");
printf("This is parent process. Writing third message into pipe\n");
write(pipeData[1], msg, BUFFERSIZE);
sprintf(msg , "University");
printf("This is parent process. Writing fourth message into pipe\n");
write(pipeData[1], msg, BUFFERSIZE);
close(pipeData[1]);
wait(NULL);
return 0;
```

}

Output :

```
(base) hiteshkumar@Hiteshs-MacBook-Pro C work % ./assign2_q10b
This is parent process. Writing first message into pipe
This is parent process. Writing second message into pipe
This is parent process. Writing third message into pipe
This is parent process. Writing fourth message into pipe
This is child process reading first message. Content is :San
This is child process reading second message. Content is :Jose
This is child process reading third message. Content is :State
This is child process reading fourth message. Content is :University
(base) hiteshkumar@Hiteshs-MacBook-Pro C work %
```

Explanation :

The child is created and the message is read by child and the parent process is used for writing message in the file.

Four message are taken - San, Jose, State, University by parent process and the messages are printed with 4 strings.

Solution 10(c)

Code :

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<signal.h>

typedef struct sigaction Sigaction;

unsigned long long size = 0;

void alarm_func(int sig){

printf("Write blocked after %llu characters. \n", size);

}
```

```
int main(){
int pipefd[2];
if(pipe(pipefd) == -1){
perror("pipe");
}
sigset_t mask , prev;
sigemptyset(&mask);
sigaddset(&mask , SIGALRM);
sigprocmask(SIG_BLOCK , &mask , &prev);
Sigaction new_action;
sigemptyset(&new_action.sa_mask);
new_action.sa_flags = SA_RESTART;
new_action.sa_handler = alarm_func;
sigaction(SIGALRM , &new_action , NULL);
sigprocmask(SIG_SETMASK , &prev, NULL);
while(1){
if(size != 0 && size % 1024 == 0){
printf("%llu characters in pipe\n", size);
}
alarm(0);
alarm(5);
write(pipefd[1], "A", sizeof(char));
size++;
}
return 0;
}
```

Explanation : The alarm(4) is set to 4 seconds and the program execution's write is blocked after 65536 characters.

Output :

```
base) hiteshkumar@Hiteshs-MacBook-Pro C work % ./q12
1024 characters in pipe
2048 characters in pipe
3072 characters in pipe
4096 characters in pipe
5120 characters in pipe
6144 characters in pipe
7168 characters in pipe
8192 characters in pipe
9216 characters in pipe
10240 characters in pipe
11264 characters in pipe
12288 characters in pipe
13312 characters in pipe
14336 characters in pipe
15360 characters in pipe
16384 characters in pipe
17408 characters in pipe
18432 characters in pipe
19456 characters in pipe
20480 characters in pipe
21504 characters in pipe
22528 characters in pipe
23552 characters in pipe
24576 characters in pipe
25600 characters in pipe
26624 characters in pipe
27648 characters in pipe
28672 characters in pipe
29696 characters in pipe
30720 characters in pipe
31744 characters in pipe
32768 characters in pipe
33792 characters in pipe
34816 characters in pipe
35840 characters in pipe
```

36864 characters in pipe
37888 characters in pipe
38912 characters in pipe
39936 characters in pipe
40960 characters in pipe
41984 characters in pipe
43008 characters in pipe
44032 characters in pipe
45056 characters in pipe
46080 characters in pipe
47104 characters in pipe
48128 characters in pipe
49152 characters in pipe
50176 characters in pipe
51200 characters in pipe
52224 characters in pipe
53248 characters in pipe
54272 characters in pipe
55296 characters in pipe
56320 characters in pipe
57344 characters in pipe
58368 characters in pipe
59392 characters in pipe
60416 characters in pipe
61440 characters in pipe
62464 characters in pipe
63488 characters in pipe
64512 characters in pipe
65536 characters in pipe
Write blocked after 65536 characters.