Roll No. : IIT2022004

OS Lab Assignment 5

Question 1

```c
// Banker's Algorithm
#include "types.h"
#include "user.h"
#include "stat.h"

int main()
{
        // P0, P1, P2, P3, P4 are the Process names here

        int n, m, i, j, k;
        n = 5; // Number of processes
        m = 3; // Number of resources
        int alloc[5][3] = { { 0, 1, 0 },
                            { 2, 0, 0 },
                            { 3, 0, 2 },
                            { 2, 1, 1 },
                            { 0, 0, 2 } };

        int max[5][3] = { { 7, 5, 3 },
                          { 3, 2, 2 },
                          { 9, 0, 2 },
                          { 2, 2, 2 },
                          { 4, 3, 3 } };

        int avail[3] = { 3, 3, 2 };

        int f[n], ans[n], ind = 0;
        for (k = 0; k < n; k++) {
                f[k] = 0;
        }
        int need[n][m];
        for (i = 0; i < n; i++) {
                for (j = 0; j < m; j++)
                        need[i][j] = max[i][j] - alloc[i][j];
        }
        int y = 0;
        for (k = 0; k < 5; k++) {
                for (i = 0; i < n; i++) {
                        if (f[i] == 0) {

                                int flag = 0;
                                for (j = 0; j < m; j++) {
                                        if (need[i][j] > avail[j]){
                                                flag = 1;
                                                break;
                                        }
                                }

                                if (flag == 0) {
                                        ans[ind++] = i;
                                        for (y = 0; y < m; y++)
                                                avail[y] += alloc[i][y];
                                        f[i] = 1;
                                }
```

```
55                          }
56                  }
57          }
58
59          int flag = 1;
60
61          for(int i=0;i<n;i++)
62          {
63          if(f[i]==0)
64          {
65                  flag=0;
66                  printf(1,"The following system is not safe");
67                  break;
68          }
69          }
70
71          if(flag==1)
72          {
73          printf(1,"Following is the SAFE Sequence\n");
74          for (i = 0; i < n - 1; i++)
75                  printf(1," P%d ->", ans[i]);
76          printf(1," P%d", ans[n - 1]);
77          }
78
79
80          return (0);
81
82 }
83
```

```
UPROGS=\
        _cat\
        _echo\
        _forktest\
        _grep\
        _init\
        _kill\
        _ln\
        _ls\
        _mkdir\
        _rm\
        _sh\
        _stressfs\
        _usertests\
        _wc\
        _zombie\
        _banker\
```

<- Made changes in the Makefile

```
SeaBIOS (version 1.15.0-1)


iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8B590+1FECB590 CA00



Booting from Hard Disk...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
$ banker
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2pid 3 banker: trap 14 err 5 on cpu 0 eip 0xffffffff a
ddr 0xffffffff--kill proc
```

Question 2

syscall.h

```
// System call numbers
#define SYS_fork    1
#define SYS_exit    2
#define SYS_wait    3
#define SYS_pipe    4
#define SYS_read    5
#define SYS_kill    6
#define SYS_exec    7
#define SYS_fstat   8
#define SYS_chdir   9
#define SYS_dup    10
#define SYS_getpid 11
#define SYS_sbrk   12
#define SYS_sleep  13
#define SYS_uptime 14
#define SYS_open   15
#define SYS_write  16
#define SYS_mknod  17
#define SYS_unlink 18
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
#define SYS_calculate_sum_of_digits  22
#define SYS_get_parent_pid  23
#define SYS_get_file_sectors 24
#define SYS_set_process_parent 25

#define SYS_change_process_queue 26
#define SYS_set_hrrn_priority 27
#define SYS_set_ptable_hrrn_priority 28
#define SYS_print_processes 29

#define SYS_sem_init 30
#define SYS_sem_acquire 31
#define SYS_sem_release 32
```

syscall.c

```
116 extern int sys_sem_init(void);
117 extern int sys_sem_acquire(void);
118 extern int sys_sem_release(void);
119
120 static int (*syscalls[])(void) = {
121 [SYS_fork]    sys_fork,
122 [SYS_exit]    sys_exit,
123 [SYS_wait]    sys_wait,
124 [SYS_pipe]    sys_pipe,
125 [SYS_read]    sys_read,
126 [SYS_kill]    sys_kill,
127 [SYS_exec]    sys_exec,
128 [SYS_fstat]   sys_fstat,
129 [SYS_chdir]   sys_chdir,
130 [SYS_dup]     sys_dup,
131 [SYS_getpid]  sys_getpid,
132 [SYS_sbrk]    sys_sbrk,
133 [SYS_sleep]   sys_sleep,
134 [SYS_uptime]  sys_uptime,
135 [SYS_open]    sys_open,
136 [SYS_write]   sys_write,
137 [SYS_mknod]   sys_mknod,
138 [SYS_unlink]  sys_unlink,
139 [SYS_link]    sys_link,
140 [SYS_mkdir]   sys_mkdir,
141 [SYS_close]   sys_close,
142 [SYS_calculate_sum_of_digits]   sys_calculate_sum_
143 [SYS_get_parent_pid]   sys_get_parent_pid,
144 [SYS_get_file_sectors] sys_get_file_sectors,
145 [SYS_set_process_parent] sys_set_process_parent,
146
147 [SYS_change_process_queue] sys_change_process_queu
148 [SYS_set_hrrn_priority] sys_set_hrrn_priority,
149 [SYS_set_ptable_hrrn_priority] sys_set_ptable_hrrr
150 [SYS_print_processes] sys_print_processes,
151
152 [SYS_sem_init] sys_sem_init,
153 [SYS_sem_acquire] sys_sem_acquire,
154 [SYS_sem_release] sys_sem_release
155 };
```

proc.c

```c
struct semaphore {
  int value;
  int locked;
  int owner;
  struct spinlock lock;
};
struct semaphore chop_stick[6];
int sem_init(int i, int v){
  acquire(&chop_stick[i].lock);
  if (chop_stick[i].locked == 0) {
    chop_stick[i].locked = 1;
    chop_stick[i].value = v;
    chop_stick[i].owner = -1;
  } else {
    release(&chop_stick[i].lock);
    return -1;
  }
  release(&chop_stick[i].lock);
  return 0;
}

int sem_acquire(int i){
  acquire(&chop_stick[i].lock);
  if (chop_stick[i].value >= 1) {
     chop_stick[i].value = chop_stick[i].value - 1;
     chop_stick[i].owner = i;
  } else {
    while (chop_stick[i].value < 1) sleep(&chop_stick[i],&chop_stick[i].lock);
    chop_stick[i].value = chop_stick[i].value - 1;
    chop_stick[i].owner = i;
  }
  release(&chop_stick[i].lock);
  return 0;
}
int sem_release(int i){
  acquire(&chop_stick[i].lock);
  chop_stick[i].value = chop_stick[i].value + 1;
  chop_stick[i].owner = -1;
  wakeup(&chop_stick[i]);
  release(&chop_stick[i].lock);
  return 0;
}
```

sysproc.c

```c
int sys_sem_init(void){
  int i , j;
  argint(0,&i);
  argint(1,&j);
  return sem_init(i,j);
}

int sys_sem_acquire(void){
  int i;
  argint(0,&i);
  return sem_acquire(i);
}

int sys_sem_release(void){
  int i;
  argint(0,&i);
  return sem_release(i);
}
```

usys.S

```
SYSCALL(sem_init)
SYSCALL(sem_acquire)
SYSCALL(sem_release)
```

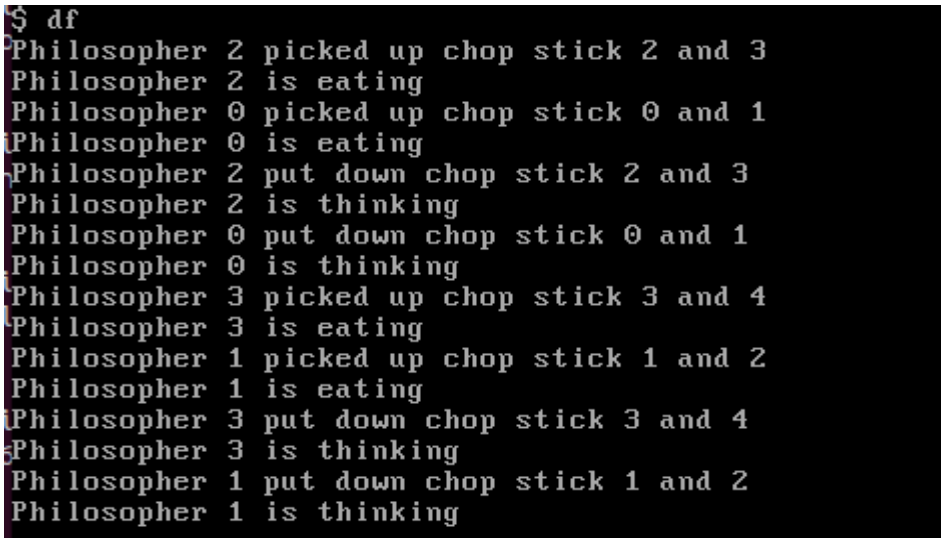
user.h

```
int sem_init(int i, int j);
int sem_acquire(int i);
int sem_release(int i);
```

Makefile

```
UPROGS=\
	_cat\
	_echo\
	_forktest\
	_grep\
	_init\
	_kill\
	_ln\
	_ls\
	_mkdir\
	_rm\
	_sh\
	_stressfs\
	_usertests\
	_wc\
	_zombie\
	_sod\
	_cpq\
	_shrrn\
	_spthrrn\
	_foo\
	_pproc\
	_philsof\
	_df\

EXTRA=\
	mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
	ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c foo.c philsof.c\
	printf.c umalloc.c cpq.c df.c shrrn.c spthrrn.c pproc.c\
	README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
	.gdbinit.tmpl gdbutil\
```

Output



Question 3

producer.c

```
#include "problem.h"

 void producer(int w)
```

```
#include "problem.h"

void consumer(int i)
{
        int n;
        MEM *S = memory();

        while(1)
        {

                sem_wait(&S->full); // Semaphore down operation
                sem_wait(&S->mutex); // Semaphore for mutual exclusion
                sem_getvalue(&S->full,&n); // Assign value of semphore full, to integer n
                printf("[CONSUMER %d ] Removed item [%d]\n",i, (S->buff)[n]);
                sem_post(&S->mutex); // Mutex up operation
                sem_post(&S->empty); // Semaphore up operation
                sleep(CONSUMER_SLEEP_SEC);

        }
}

int main(int argc,char *argv[])
{
        consumer(argv[1]-"0");
        return 0;
}
```

```
    init();
    producer(argv[1][0]-'0');
    return 0;
}
```

consumer.c

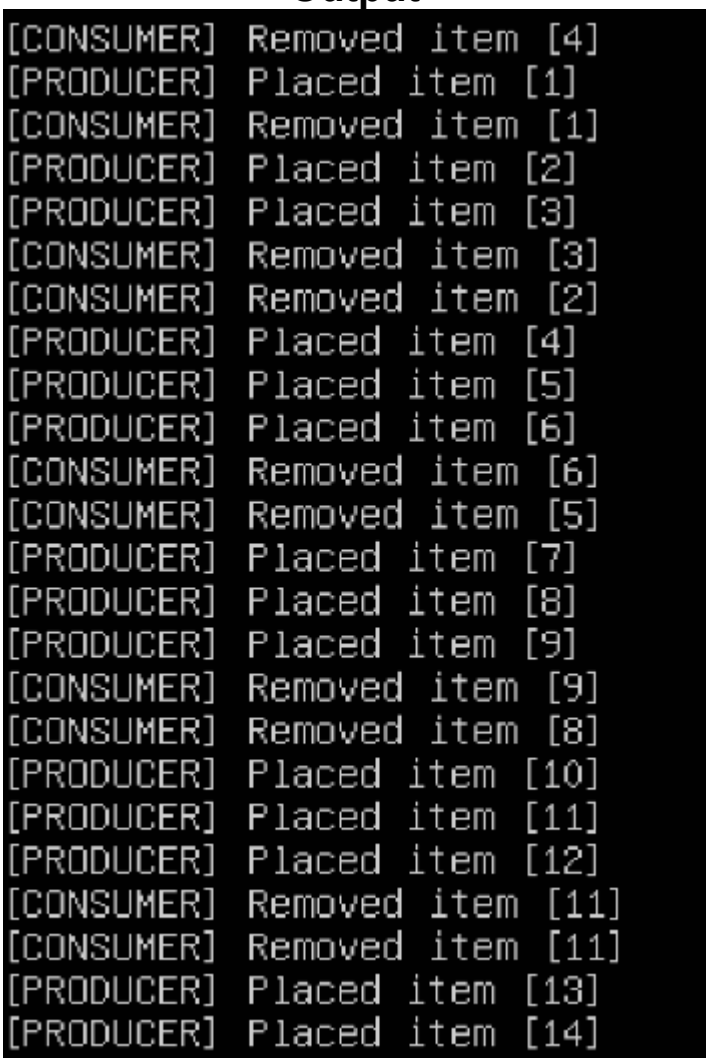
problem.h

```c
#include <stdio.h>
#include <semaphore.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <stdlib.h>
#define BUFFER_SIZE 10
#define CONSUMER_SLEEP_SEC 3
#define PRODUCER_SLEEP_SEC 1
#define KEY 1010

// A structure to store BUFER and semaphores for synchronization
typedef struct
{
        int buff[BUFFER_SIZE];
        sem_t mutex, empty, full;

} MEM;

// Method for shared memory allocation
MEM *memory()
{
        key_t key = KEY;
        int shmid;
        shmid = shmget(key, sizeof(MEM), IPC_CREAT | 0666);
        return (MEM *) shmat(shmid, NULL, 0);
}

void init()
{
        // Initialize structure pointer with shared memory
        MEM *M = memory();

        // Initialize semaphores
        sem_init(&M->mutex,1,1);
        sem_init(&M->empty,1,BUFFER_SIZE);
        sem_init(&M->full,1,0);
}
```

problem_sol.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "problem.h"
#include <unistd.h>
int main()
{
        int num_prod=3,num_con=2;
        int pi=fork();
        if(pi==0){
        char * args[]={"./producer","0"};
        execvp("./producer",args);
                for(int i=0;i<num_prod-1;i++){
                        if(fork()){
                                char *args[]={"./producer","0"+i+1};
                                 execvp("./producer",args);
                        }
                        exit(1);
                }
        }
        else{
            for(int i=0;i<num_con;i++){
                if(fork()){
                    char *args[]={"./consumer","0"+i};
                    execvp("./consumer",args);
                    exit(1);
                }
            }
        }
}
```

Output


[CONSUMER] Removed item [4]
[PRODUCER] Placed item [1]
[CONSUMER] Removed item [1]
[PRODUCER] Placed item [2]
[PRODUCER] Placed item [3]
[CONSUMER] Removed item [3]
[CONSUMER] Removed item [2]
[PRODUCER] Placed item [4]
[PRODUCER] Placed item [5]
[PRODUCER] Placed item [6]
[CONSUMER] Removed item [6]
[CONSUMER] Removed item [5]
[PRODUCER] Placed item [7]
[PRODUCER] Placed item [8]
[PRODUCER] Placed item [9]
[CONSUMER] Removed item [9]
[CONSUMER] Removed item [8]
[PRODUCER] Placed item [10]
[PRODUCER] Placed item [11]
[PRODUCER] Placed item [12]
[CONSUMER] Removed item [11]
[CONSUMER] Removed item [11]
[PRODUCER] Placed item [13]
[PRODUCER] Placed item [14]