

```
/******  
/*** ~fredm/308/files/call_help_asn4.txt ***/  
/*** from Prof. Moloney ***/  
/*****
```

```
/*** need a .h file with content as shown below ***/
```

```
#include <time.h>  
#include <sys/types.h>  
#include <sys/ipc.h>  
#include <sys/sem.h>  
#include <sys/shm.h>  
#include <sys/signal.h>  
  
#define SEMKEY (key_t)5763  
#define MEMKEY (key_t)5763  
#define NUMFLAVORS 4  
#define NUMSLOTS 50  
#define NUMSEMIDS 3  
#define PROD 0  
#define CONSUMER 1  
#define OUTPTR 2  
  
struct donut_ring{  
    int flavor [NUMFLAVORS] [NUMSLOTS];  
    int outptr [NUMFLAVORS];  
};  
  
extern int p (int, int);  
extern int v (int, int);  
extern int semsetall (int, int, int);  
  
/*** a .c file for the producer's code ***/  
  
int shmids, semids[3];  
void sig_handler (int);  
  
int main(int argc, char *argv[])  
{  
    int in_ptr [NUMFLAVORS];  
    int serial [NUMFLAVORS];  
    int i,j,k;  
    struct donut_ring *shared_ring;  
    struct timeval randtime;  
    /* producer initializes serial counters and in-pointers */  
    for(i = 0; i < NUMFLAVORS; i++){  
        in_ptr [i] = 0;  
        serial [i] = 0;  
    }  
    /** begin syscall signal comment for signal handling  
    #include <signal.h>  
  
    int sigaction(int sig, struct sigaction *new_action,  
                  struct sigaction *old_action);  
  
    struct sigaction{  
        void (*sa_handler)();  
        sigset_t sa_mask;
```

```
    int          sa_flags;
};

sigemptyset      (sigset_t *mask);
sigfillset       (sigset_t *mask);
sigaddset        (sigset_t *mask, int SIGNAL);
sigdelset        (sigset_t *mask, int SIGNAL);

    SIGHUP        1          hangup
    SIGINT        2          interrupt
    SIGQUIT       3*        quit
    SIGILL        4*        illegal instruction
    SIGTRAP       5*        trace trap
    SIGABRT       6*        abort (generated by abort(3) routine)
    SIGEMT        7*        emulator trap
    SIGFPE        8*        arithmetic exception
    SIGKILL       9          kill (cannot be caught, blocked, or ignored)
    SIGBUS        10*       bus error
    SIGSEGV       11*       segmentation violation
    SIGSYS        12*       bad argument to system call
    SIGPIPE       13        write on a pipe with no reader
    SIGALRM       14        alarm clock
    SIGTERM       15        software termination signal

***** end syscall signal comment *****/

/***** only need to catch some signals *****/

void          sig_handler ( int ); /* declare signal handler function */
sigset_t      mask_sigs;
int           i, nsigs;
int           sigs [] = {SIGHUP, SIGINT, SIGQUIT, SIGBUS,
                        SIGTERM, SIGSEGV, SIGFPE};

nsigs = sizeof (sigs) / sizeof (int)
sigemptyset (&mask_sigs);
for(i=0; i< nsigs; i++)
    sigaddset (&mask_sigs, sigs [i]);
for(i = 0; i < nsigs; i++){
    new.sa_handler = sig_handler;
    new.sa_mask    = mask_sigs;
    new.sa_flags   = 0;
    if(sigaction (sigs [i], &new, NULL) == -1){
        perror("can't set signals: ");
        exit(1);
    }
}
}

/**** begin syscall shmget comment for shared memory
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int shmget(key_t key, size_t size, int shmflg)
```

```
***** end syscall shmget comment *****/

    if((shmid = shmget (SHMKEY, sizeof(struct donut_ring),
                        IPC_CREAT | 0600)) == -1){
        perror("shared get failed: ");
        exit(1);
    }

/**/ begin syscall shmat comment for shared memory
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

void *shmat (int shmid, const void *attach_addr,
             int shmflg);

***** end syscall shmat comment *****/

    if((shared_ring = (struct donut_ring *)
        shmatt (shmid, (const void *)0, 0)) == -1){
        perror("shared attach failed: ");
        sig_handler(-1);
    }

/**/ begin syscall semget comment for semaphores
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

int semget (key_t key, int nsems, int semflg);

***** end syscall semget comment *****/

    for(i=0; i<NUMSEMIDS; i++)
        if ((semid[i] = semget (SEMKEY+i, NUMFLAVORS,
                                IPC_CREAT | 0600)) == -1){
            perror("semaphore allocation failed: ");
            sig_handler(-1);
        }

/**/ begin syscall gettimeofday comment for time seed
#include <sys/time.h>

int gettimeofday (struct timeval *tp, struct timezone *tzp);

struct timeval {
    long    tv_sec;           /* seconds
    long    tv_usec;        /* and microseconds
}

***** end syscall gettimeofday comment *****/
```

```
    gettimeofday (&randtime, (struct timezone *)0);

/* use microsecond component for uniqueness */

    unsigned short xsubl[3];
    xsubl[0] = (ushort) randtime.tv_usec;
    xsubl[1] = (ushort)(randtime.tv_usec >> 16);
    xsubl[2] = (ushort)(getpid());

/* use nrand48 with xsubl to get 32 bit random number */

    j=nrand48(xsubl) & 3;

/* use the semsetall utility to set initial semaphore values */

    if(semsetall (semid [PROD],
                  NUMFLAVORS, NUMSLOTS) == -1){
        perror("semsetsall failed: ");
        sig_handler(-1);
    }
    if(semsetall (semid [CONSUMER],
                  NUMFLAVORS, 0) == -1){
        perror("semsetsall failed: ");
        sig_handler(-1);
    }
    if(semsetall (semid [OUTPTR],
                  NUMFLAVORS, 1) == -1){
        perror("semsetsall failed: ")
        sig_handler(-1);
    }
/* the rest of the producer/consumer code follows, */
/* including the producer signal handler below */

void    sig_handler(int sig)
{
    int    i,j,k;

    printf("In signal handler with signal # %d\n",sig);

    if(shmctl(shmid, IPC_RMID, 0) == -1){
        perror("handler failed shm RMID: ");
    }
    for(i = 0; i < NUMSEMIDS; i++){
        if(semctl (semid[i], 0,
                  IPC_RMID, (union semun)0) == -1){
            perror("handler failed sem RMID: ");
        }
    }
    exit(5);
}

/*****
*** a utility .c file should include the following *****/
```

```
/** semaphore utilities: */

int p (int semidgroup, int donut_type)
{
    struct sembuf semopbuf; /** struct in <sys/sem.h> */

/** begin syscall semop comment for semaphore operations

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

int semop (int semid, struct sembuf *sops, int nsops);
struct sembuf {
    short  sem_num;          /* semaphore index
    short  sem_op;          /* semaphore operation
    short  sem_flg;         /* operation flags
};

***** end syscall semop comment *****/

    semopbuf.sem_num = donut_type;
    semopbuf.sem_op = (-1); /** -1 is a P operation */
    semopbuf.sem_flg = 0;

    if(semop (semidgroup, &semopbuf,1) == -1){
        perror ("p operation failed: ");
        return (-1);
    }
    return (0);
}

int v (int semidgroup, int donut_type)
{
    struct sembuf semopbuf;

    semopbuf.sem_num = donut_type;
    semopbuf.sem_op = (+1); /** +1 is a V operation */
    semopbuf.sem_flg = 0;

    if(semop(semidgroup, &semopbuf,1) == -1){
        perror("p operation failed: ");
        return (-1);
    }
    return (0);
}

int semsetall (int semgroup, int number_in_group,
               int set_all_value)
{
    int i, j, k;
    union semun /** need this union */
    {
        int val;
    }
}
```

```
    struct semid_ds *buf;
    unsigned short int *array;
    struct seminfo *__buf;
}sem_ctl_un;
```

```
/** begin syscall semctl comment for semaphore control
```

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
int semctl (int semid, int semnum, int cmd, ...);
```

The fourth argument is optional and depends on the operation requested. If required, it is of the type union semun, which the application program must explicitly declare as follows:

The user should define a union like the following to use for command specific values for 'semctl' in the fourth argument position ... when used this argument is needed, the union is passed by value and the semctl call selects the appropriate member (based on the command argument)

```
union semun
{
    int val;                <= value for SETVAL
    struct semid_ds *buf;   <= buffer for IPC_STAT & IPC_SET
    unsigned short int *array; <= array for GETALL & SETALL
    struct seminfo *__buf;  <= buffer for IPC_INFO
};
```

Previous versions of <sys/sem.h> used to define this union but this is incorrect. One can test the macro _SEM_SEMUN_UNDEFINED to see whether one must define the union or not, but I will define it myself for this example function (semsetall)

```
***** end syscall semctl comment *****/
```

```
    sem_ctl_un.val = set_all_value; /** for command SETVAL ***/
    for (i = 0; i < number_in_group; i++){
        if(semctl(semgroup, i, SETVAL, sem_ctl_un) == -1){
            perror("semset failed");
            return (-1);
        }
    }
    return (0);
}
```