Introduction to System Programming

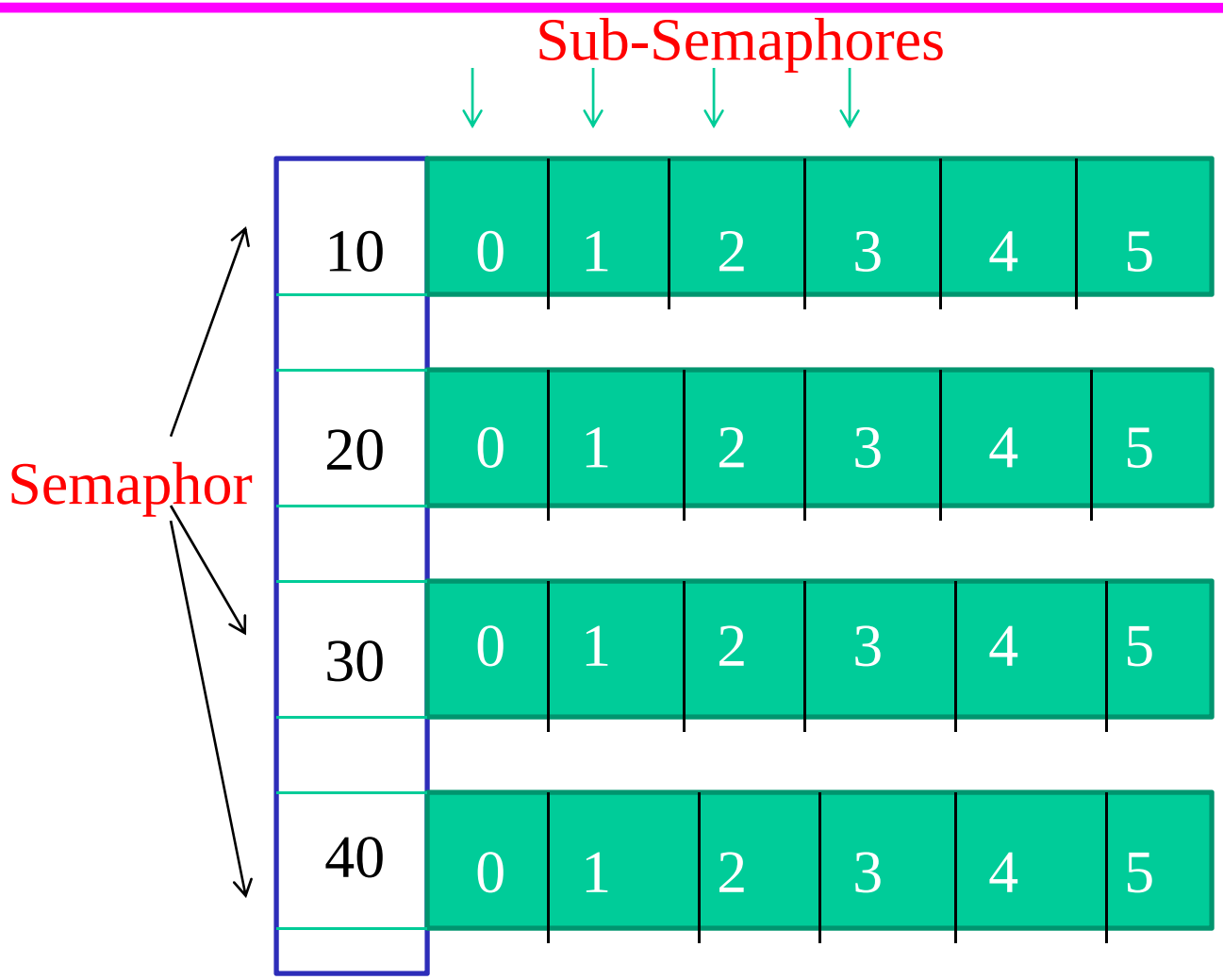
System Calls For Semaphore Management

Minal Shah

Outline

- System Calls for Semaphore Management
  - semget
  - semctl
  - semop
  - fsync

Semaphore structure



semget System Call

- Semget Function : To create a semaphore, or gain access to one that exists.
- Include: <sys/types.h> <sys/ipc.h> <sys/sem.h>
- Command: ***int semget (key_t key, int nsems, int semflg);***
  Arguments
  - ***key_t key***: used to identify a semaphore set
  - ***int nsems***: the number of semaphores in the set.
  - ***int semflg***: specify access permissions and/or special creation condition(s).
- Returns: Success: the semaphore identifier (semid);
  Failure :-1; Sets errno: yes
- To see whether semaphore is created use :

```
$ ipcs -s
ID Key   mode    Owner   nsems
```

Creating and Accessing Semaphore Sets

```
main()
{
    key=(key_t)0x20;
    nsem=1
    semid=semget(key, nsem, IPC_CREAT|0666)

}
```

Read-alter mode

Flag: IPC_EXCL: Exclusive creation of semaphore

IPC_CREAT|0666|IPC_EXCL

Program to create semaphore (S1.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
void main()
{
    int semid,key,nsem;
    key=(key_t)0x20;
    nsem=0;
    semid=semget(key,nsem,IPC_CREAT|0666);
    printf("created semaphore with id :%d\n",semid);
}
```

```
$./a.out
created semaphore
   with id :-1
```

To see whether semaphore is created or not

//command to verify the creation of semaphore

$ipcs -s

------ Semaphore Arrays --------

key semid owner perms nsems

Program to create semaphore (S2.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
void main()
{
    int semid,key,flag,nsem;
    key=(key_t)0x20;
    flag=IPC_CREAT|0666;
    nsem=1;
    semid=semget(key,nsem,flag);
    printf("created semaphore with id :%d\n",semid);
}
```

Program to create semaphore (S2.txt)

```
$./a.out
created semaphore with id :0


$ipcs –s
------ Semaphore Arrays --------
   key              semid      owner      perms      nsems
   0x00000020     0          student    666          1
```

To find maximum number of semaphore sets available in linux system (s3.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<stdlib.h>
void main()
{
    int semid,nsemset,key,flag,nsem;
    nsem=1;
    flag=IPC_CREAT|0666;
    for(nsemset=0;;nsemset++)
    {
        key=(key_t)nsemset;
        semid=semget(nsemset,nsem,flag);
```

To find maximum number of semaphore sets available in linux system (s3.c)

```
if(semid > 0)
        printf("created semaphore with id :%d\n",semid);
    else
    {
        printf("Maximum number of semaphore set are  %d
                \n",nsemset);
        exit(0);
    }
 }
}
```

To find maximum number of semaphore sets available in linux system (S3.txt)

```
$./a.out
created semaphore with id :32769
created semaphore with id :65538
created semaphore with id :98307
created semaphore with id :131076
created semaphore with id :163845
created semaphore with id :196614
created semaphore with id :229383
created semaphore with id :262152
created semaphore with id :294921
created semaphore with id :327690
created semaphore with id :360459
created semaphore with id :393228
created semaphore with id :425997
```

To find maximum number of semaphore sets available in linux system (S3.txt)

created semaphore with id :458766
created semaphore with id :491535
created semaphore with id :524304
created semaphore with id :557073
created semaphore with id :589842
created semaphore with id :622611
created semaphore with id :655380
created semaphore with id :688149
created semaphore with id :720918
created semaphore with id :753687
created semaphore with id :786456
created semaphore with id :819225
created semaphore with id :851994
created semaphore with id :884763

To find maximum number of semaphore sets available in linux system (S3.txt)

created semaphore with id :917532
created semaphore with id :950301
created semaphore with id :983070
created semaphore with id :1015839
created semaphore with id :1048608
Maximum number of semaphore set are 32

When issued : ($ipcs –s)

```
------ Semaphore Arrays --------
key         semid      owner      perms      nsems
0x00000020  0          student    666        1
0x00000000  32769      student    666        1
0x00000001  65538      student    666        1
0x00000004  163845     student    666        1
0x00000005  196614     student    666        1
0x00000002  98307      student    666        1
0x00000003  131076     student    666        1
0x00000006  229383     student    666        1
0x00000007  262152     student    666        1
0x00000008  294921     student    666        1
0x00000009  327690     student    666        1
0x0000000a  360459     student    666        1
0x0000000b  393228     student    666        1
0x0000000c  425997     student    666        1
```

When issued : ($ipcs –s)

0x0000000d	458766	student	666	1
0x0000000e	491535	student	666	1
0x0000000f	524304	student	666	1
0x00000010	557073	student	666	1
0x00000011	589842	student	666	1
0x00000012	622611	student	666	1
0x00000013	655380	student	666	1
0x00000014	688149	student	666	1
0x00000015	720918	student	666	1
0x00000016	753687	student	666	1
0x00000017	786456	student	666	1
0x00000018	819225	student	666	1
0x00000019	851994	student	666	1
0x0000001a	884763	student	666	1
0x0000001b	917532	student	666	1
0x0000001c	950301	student	666	1
0x0000001d	983070	student	666	1
0x0000001e	1015839	student	666	1
0x0000001f	1048608	student	666	1

To find maximum number of semaphore in each set

(s4.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<stdlib.h>
void main()
{
    int semid,i,key,flag,nsem;
    key=(key_t)0x30;
    nsem=1;
    flag=IPC_CREAT|0666;
    for(i=0;;i++)
    {
        nsem=i+1;
        semid=semget(key,nsem,flag);
```

To find maximum number of semaphore in each set

(s4.c)

```
    if(semid > 0)
            printf("created semaphore with id :%d\n",semid);
        else
        {
            printf("Maximum number of semaphore set are %d\n",i);
            exit(0);
        }
        semctl(semid,0,IPC_RMID,0);
    }
}
```

To find maximum number of semaphore in each set (S4.txt)

$./a.out
created semaphore with id :1081377
created semaphore with id :1114145
created semaphore with id :1146913
created semaphore with id :1179681
created semaphore with id :1212449
created semaphore with id :1245217
● ● ●
Maximum number of semaphore set are 32000

Semaphore exclusivity (s7.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<stdlib.h>
void main()
{
    int semid,key,flag,nsem,i;
    for(i=0;i<2;i++)
    {
        key = (key_t)0x30;
        flag=IPC_CREAT|0666|IPC_EXCL;
        nsem=1;
```

Semaphore exclusivity (s7.c)

```
        semid=semget(key,nsem,flag);
        if(semid > 0)
            printf("created semaphore with id :%d \n",
                        semid);
        else
            perror("semget");
    }
}
```

Semaphore exclusivity (s7.txt)

- Note: delete the semaphore with KEY 30 before running the above program.
- To do this

- $ipcrm –S 0x30
- $./a.out

created semaphore with id :999424031

semctl System Call - Semaphore Control

- Function :To perform a variety of generalized control operations on the system semaphore structure, on the semaphores as a set and on individual semaphores.
- Include <sys/tyoes.h> <sys/ipc.h> <sys/sem.h>
- Command: ***int semctl (int semid, int semum, int cmd, /* union semun arg*/ ...);***
- ***Returns :*** **Success :** 0 or the value requested
  Failure : -1
  Seterrno : - Yes

Arguments

- ***int semid***: a valid semaphore identifier.
- ***int semum***: the number of semaphores in the semaphore set.
- ***int cmd***: an integer command value (IPC_STAT, IPC_SET, ..).
- ***arg***: union of type ***semun***.

semctl System Call - Semaphore Control

- The semctl system call takes four arguments:
  - The first argument, semid, is a valid semaphore identifier that was returned by a previous semget system call.
  - The second argument, semnum, is the number of semaphores in the semaphore set.
  - The third argument to semctl, cmd, is an integer command value. The cmd value directs semctl to take one of several control actions. Each action requires specific access permissions to the semaphore control structure.

cmd values of *semctl* Call

- IPC_STAT: return the current values of the semid_ds structure for the indicated semaphore identifier.
- IPC_SET: modify a restricted number of members in the semid_ds structure.
- IPC_RMID: remove the semaphore set.
- GETAll: return the current values of the semaphore set.
- SETALL: Initialize all semaphores in a set to the values stored in the array referenced by the fourth arguments to semctl.
- GETVAL: return the current of the individual semaphore referenced by the value of the ***semnum*** argument.
- SETVAL: set the value of a single semaphore in a set

.....

semctl System Call - Semaphore Control

- The semctl system call takes four arguments:
- The fourth argument to semctl, arg, is a union of type semun. Given the action specified by the preceding cmd argument, the data in arg can be one of any of the following four values:
  - An integer used with SETVAL to indicate a specific value for a particular semaphore within the semaphore set.
  - A reference to a semid_ds structure where information is returned when IPC_STAT or IPC_SET is specified.
  - A reference to an array of type unsigned short integers; the array is used either to initialize the semaphore set or as a return location when specifying GETALL.
  - A reference to a seminfo structure when IPC_INFO is requested.

union *semun* in the *semctl* call

A ***union*** is is a later-day version of the Pascal *variant record*. It is a data structure that can take on multiple forms.
semctl() requires a union to handle the different kinds of data that can be provided to it or received from it.

```
union  semun {
        int             val;
        struct          semid_ds  *buf;
        ushort          * array;
} arg; // declares a semun named arg
```

The value in arg is one of:

- int val: an integer (0 or others),
- struct semid_ds *buf: a reference to a semid_ds structure,
- ushort *array: the base address of an array of short integers (the values for the semaphore(s)).

Getting and setting Semaphore values (s8.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
#include<stdlib.h>
#include<errno.h>
void main()
{
    int semid,retval;
    semid=semget(0x20,1,0666|IPC_CREAT);
    retval=semctl(semid,0,GETVAL,0);
    printf("value returned is %d\n",retval);
}
```

- $./a.out

value returned is 0

Setting different value to Semaphore (s9.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
#include<stdlib.h>
#include<errno.h>
```

Setting different value to Semaphore (s9.c)

```
void main()
{
    int semid,retval;
    semid=semget(0x20,1,0666|IPC_CREAT);
    semctl(semid,0,SETVAL,1);
    retval=semctl(semid,0,GETVAL,0);
    printf("value of the semaphore after setting is %d\n",retval);
    semctl(semid,0,SETVAL,2);
    retval=semctl(semid,0,GETVAL,0);
    printf("value of the semaphore after setting is %d\n",retval);
  }
```

- $./a.out

value of the semaphore after setting is 1

value of the semaphore after setting is 2

Who is using Resources (s10.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
#include<errno.h>
void main()
{
    int semid,retval;
    semid=semget(0x20,1,0666|IPC_CREAT);
    semctl(semid,0,SETVAL,1);
    retval=semctl(semid,0,GETPID,0);
    printf("PID returned by semctl is %d and actual PID is
   %d\n",retval,getpid());
}
```

- $./a.out
- PID returned by semctl is 3409 and actual PID is 3409
- Note kill the semaphore first and then run this program twice

Who is using Resources . Here subsemaphore is intialized(s11.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
#include<errno.h>
void main()
{
    int semid,retval;
    semid=semget(0x20,1,0666|IPC_CREAT);
    retval=semctl(semid,0,GETPID,0);
    printf("PID returned by semctl is %d and actual PID is
            %d\n",retval,getpid());
    retval=semctl(semid,0,SETVAL,1);
}
```

- Note : remove the semaphore before running the program
- $./a.out
- PID returned by semctl is 3409 and actual PID is 3429

Semaphore structure

Sem_ids

```
/* One sem_array data structure for each set of semaphores in the system. */

struct sem_array {
struct kern_ipc_perm     sem_perm;  /* permissions .. see ipc.h */
time_t          sem_otime; /* last semop time */
time_t          sem_ctime; /* last change time */
struct sem   *sem_base;/* ptr to first semaphore in array */
struct sem_queue *sem_pending;     /* pending operations to be processed */
struct sem_queue    **sem_pending_last; /* last pending operation */
struct sem_undo             *undo;         /* undo requests on this array */
unsigned long          sem_nsems;     /* no. of semaphores in array */
};
```

semid_ds

```
struct ipc_perm
{
  key_t  key;
  ushort uid;   /* owner euid and egid */
  ushort gid;
  ushort cuid;  /* creator euid and egid */
  ushort cgid;
  ushort mode;  /* access modes see mode flags below */
  ushort seq;   /* slot usage sequence number */
};
```

```
struct sem {
        u_short semval;
        short   sempid;
        u_short semncnt;
        u_short semzcnt;
    };
```

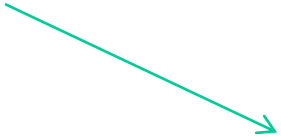
Waiting for positive

Waiting for zero

Sembuf

```
struct sembuf
{
        ushort sem_num;
        short      sem_op;
        short sem_flg;
}
```

Sub semaphore



0, IPC_NOWAIT, SEM_UNDO

int semop(int semid, struct sembuf *sops, unsigned nsops);

semop Call - Semaphore Operation

- Function : to perform operations on individual semaphores.
- Include: <sys/types.h> <sys/ipc.h> <sys/sem.h>
- Command: ***int semop (int semid, struct sembuf *sops, size_t nsops);***
- Returns: Success: 0; Failure; -1; Sets errno: Yes.
- Arguments
  - ***int semid***: semaphore identifier.
  - ***struct sembuf *sops***: a reference to the address of an array of semaphore operations that will be performed on the semaphore set denoted by the semid value.
  - ***size_t nsops***: the number of elements in the array of semaphore operations.

Actions Taken by *semop*

- If ***semop*** value:
  - is positive: : Add ***sem_op*** to ***semval***. This is a release of a resource
  - is zero: : The caller will block until the semaphore's value becomes zero.
  - is negative: The caller is blocked until the semaphore's value (***semval***) becomes greater than or equal to the absolute value of ***sem_op***. Then, the absolute value of ***sem_op is subtracted*** from ***semval***.

SEMOP semaphore (y.c)

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
void main()
{
    int semid,pid;
    struct sembuf sop;
    semid=semget(0x20,1,0666|IPC_CREAT);
    pid = fork();
```

SEMOP semaphore (y.c)

```
if(pid == 0)
 {
     sleep(2);
     printf("Child before semop\n");
     sop.sem_num = 0;
     sop.sem_op = 0;
     sop.sem_flg = 0;
     semop(semid,&sop,1);
     printf("child over\n");
 }
```

SEMOP semaphore (y.c)

```
    else
     {
          printf("parent before 1st semctl\n");
          semctl(semid,0,SETVAL,1);
          printf("Parent sleeping\n");
          sleep(5);
          printf("parent before 2nd semctl\n");
          semctl(semid,0,SETVAL,0);
          printf("Parent over\n");
     }
}
```

Running the previous program

- parent before 1st semctl
  Parent sleeping
  parent before 2nd semctl
  Parent over
  Child before semop
  child over

  ------ Semaphore Arrays --------
  key semid owner perms nsems
  0x00000020 32768 student 666 1

Sturcture of semaphore

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
void main()
{
    int semid;
    struct semid_ds status;
    semid=semget((key_t)0x20,10,0666|IPC_CREAT);
    semctl(semid,0,IPC_STAT,&status);
    printf("No of semphoares in set are %ld\n",status.sem_nsems);
    printf("My user id is %u\n",getuid());
    printf("Owner user id is %u\n",status.sem_perm.uid);
    printf("My group id is %u\n",getgid());
```

Sturcture of semaphore

```
    printf("Owner group id is %u\n",status.sem_perm.gid);
    printf("Creator user id is %u\n",status.sem_perm.cuid);
    printf("Creator group id is %u\n",status.sem_perm.cgid);
    printf("Access mode is %o\n",status.sem_perm.mode);
}
```

```
No of semphoares in set are 94610192697616
My user id is 1001
Owner user id is 0
My group id is 1001
Owner group id is 1970169159
Creator user id is 0
Creator group id is 9
Access mode is 0
```

fsync, fdatasync - synchronize a file's in-core state with storage device

- **fsync**() transfers ("flushes") all modified in-core data of (i.e., modified buffer cache pages for) the file referred to by the file descriptor *fd* to the disk device (or other permanent storage device) where that file resides.
- The call blocks until the device reports that the transfer has completed.
- It also flushes metadata information associated with the file (see **stat**(2)).
- Calling fsync() does not necessarily ensure that the entry in the directory containing the file has also reached disk.

fsync, fdatasync - synchronize a file's in-core state with storage device

- Include : **<unistd.h>**
- Command : **int fsync(int *fd*);**
  int fdatasync(int *fd*);
- Return : On success, zero is returned.
  On error, -1 is returned,
  and *errno* is set appropriately.

Program for fsync()

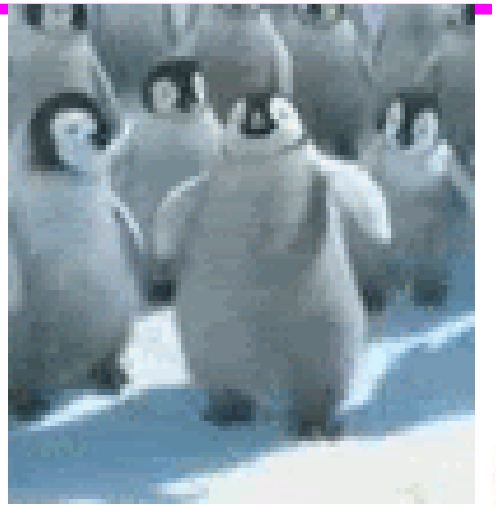
```
#include <stdio.h>
#include <fcntl.h>
int main()
{
   char my_write_str[] = "1234567890";
   char my_read_str[100];
   char my_filename[] = “m1.txt";
   int my_file_descriptor, close_err;
    /* Open the file. */
   my_file_descriptor = open (my_filename, O_RDWR |
   O_CREAT | O_TRUNC);
```

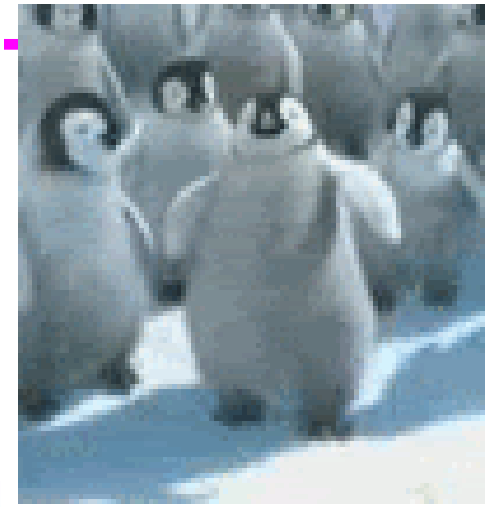
Program for fsync()

```
/* Write 10 bytes of data and make sure it's written */
write (my_file_descriptor, (void *) my_write_str, 10);
fsync (my_file_descriptor);

/* Seek the beginning of the file */
lseek (my_file_descriptor, 0, SEEK_SET);

 /* Read 10 bytes of data */
read (my_file_descriptor, (void *) my_read_str, 10);
/* Terminate the data we've read with a null character
*/
my_read_str[10] = '\0';
```

Program for fsync()

```
    printf ("String read = %s.\n", my_read_str);
    close (my_file_descriptor);
    return 0;
}
```





Minal Shah