Module 10

Shared Memory

Overview

- Objectives
- Relevance

Memory Mapping

```
1   #include <sys/types.h>
2   #include <sys/mman.h>
3   #include <sys/stat.h>
4   #include <fcntl.h>
5   #include <stdio.h>
6   #include <stdlib.h>
7   #include <unistd.h>
8
9   main(int argc, char *argv[]) {
10
11    int fd;
12    caddr_t addr;
13    struct stat statbuf;
14
15    if( argc != 2 ) {
16      fprintf(stderr,"Usage: mycat2 filename\n");
17      exit(1);
18    }
19
```



```
20    if( stat(argv[1], &statbuf) == -1 ) {
21      perror("stat");
22      exit(1);
23    }
24
25    fd = open( argv[1], O_RDONLY );
26
27    if (fd == -1) {
28      perror("open");
29      exit(1);
30    }
31
32    addr = mmap((caddr_t)NULL, statbuf.st_size,
33                PROT_READ, MAP_SHARED, fd, (off_t)0);
34
35    if (addr == MAP_FAILED {
36      perror("mmap");
37      exit(1);
38    }
39
40    /* no longer need fd */
41    close(fd);
```

```
42
43    write(1, addr, statbuf.st_size);
44    return(0);
45  }
```


Sizing a File

```
1   #include <sys/types.h>
2   #include <sys/stat.h>
3   #include <sys/mman.h>
4   #include <fcntl.h>
5   #include <stdio.h>
6   #include <unistd.h>
7
8   main() {
9
10    int fd;
11    int pagesize = sysconf(_SC_PAGESIZE);
12    caddr_t addr;
13
14    printf("Page size is %d bytes.\n",pagesize);
15    fd = open
16      ("mapfile", O_RDWR | O_CREAT | O_TRUNC, 0666);
17
18    if (fd == -1) {
19      perror("open");
```



```
20      exit(1);
21    }
22
23    if (ftruncate(fd, (off_t)(6 * pagesize)) == -1) {
24      perror("ftruncate");
25      exit(1);
26    }
27
28    system("ls -l mapfile");
29
30    addr = mmap(NULL, 6*pagesize, PROT_READ |
31        PROT_WRITE, MAP_SHARED, fd, (off_t)0);
32
33    if (addr == MAP_FAILED {
34      perror("mmap");
35      exit(1);
36    }
37
38    /* no longer need fd */
39    close(fd);
40
41    /* Write into file! */
```

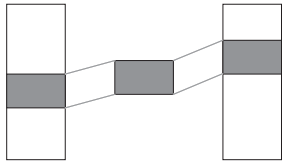


```
42    (void)strcpy(addr, "Test string.\n");
43
44    /* Display 1 line of mapfile */
45    system("head -1 mapfile");
46    return(0);
47  }
```

System V Shared Memory

- `shmget()` – Gets the shared memory identifier
- `shmat()` – Attaches the shared memory segment
- `shmctl()` – Provides shared memory control operations
- `shmdt()` – Detaches from data segment of the calling process

System V Shared Memory



Example 1

```
1   struct something *shmaddr;
2   int shmid;
3   key_t key = ftok("/etc/passwd", 'J');
4   .....
5   shmid = shmget(key, sizeof(struct something),
6       IPC_CREAT | 0666);
7   if (shmid == -1)
8     /* Handle error. */
9
10  shmaddr = (struct something *)shmat(
11      shmid, NULL, 0);
12  if (shmaddr == (struct something *)-1)
13    /* Handle error. */
14  .....
15  /* Use the shared memory here */
16  .....
17  shmdt(shmaddr);
18  shmctl(shmid, IPC_RMID, (struct shmid_ds *)NULL);
```


Example 2

```
1   #define NUM_ELEMENTS  50
2   .....
3   struct something *shmaddr;
4   int  shmid;
5
6   shmid = shmget(key,
7       NUM_ELEMENTS*sizeof(struct something),
8       IPC_CREAT | 0666);
9   if (shmid == -1)
10    /* Handle error */
11  shmaddr = (struct something *)shmat(
12      shmid, (char *)NULL, 0);
13  if (shmaddr == (struct something *)-1)
14    /* Handle error */
15  .....
16  /* May now use shmaddr[0], shmaddr[1],
17  .... shmaddr[49] */
18  .....
19  shmdt(shmaddr);
20  shmctl(shmid, IPC_RMID, (struct shmid_ds *)NULL);
```

POSIX Shared Memory

Purpose	System V	POSIX
Open connection	`shmget()`	`shm_open()`
Set size	`shmget()`	`ftruncate()`
Attach	`shmat()`	`mmap()`
Detach	`shmdt()`	`munmap()`
Remove rendezvous	`shmctl()` with `IPC_RMID`	`shm_unlink()`

pos_shm_play.c

```
1   /* pos_shm_play : set up an area of shared memory,
2     or attach to an existing one. Simply report what
3     is there, and write the process id. Wait a random
4     period and repeat. */
5
6   #define _POSIX_SHARED_MEMORY_OBJECTS
7
8   #include <sys/types.h>
9   #include <sys/mman.h>
10  #include <fcntl.h>
11
12  #define DIE(x) perror(x),exit(1)
13  #define MEMSIZ 256
14
15  main(int argc, char **argv) {
16    int shmid, delay;
17    char * path = "/SHM_DEMO";
18    char * shm;
19
20    /* Delete segment if -u on commandline. */
```



```
21    if (argc == 2 && strcmp(argv[1],"-u") == 0) {
22      if (shm_unlink(path) == -1)
23        DIE("Failed to unlink shared memory");
24        exit(0);
25    }
26
27    if ((shmid = shm_open(path,O_RDWR)) == -1) {
28      printf("Creating new shared memory segment. \n");
29      if ((shmid = shm_open(path,O_RDWR|O_CREAT,0666))
30          == -1)
31        DIE("Failed to create shm");
32      if (ftruncate(shmid,MEMSIZ) == -1)
33        DIE("Setting the size failed");
34    }
35
36    if ((shm = mmap(NULL, MEMSIZ,
37        PROT_READ|PROT_WRITE,
38        MAP_SHARED,shmid,0)) == MAP_FAILED)
39      DIE("mmap failed");
40
41    srand(getpid());
42    delay = rand() % 5 ;
```



```
43
44    for (;;) {
45      printf("Shared memory contains : %s\n",shm);
46      sprintf(shm,"Process %d was here\n",getpid());
47      sleep(delay);
48    }
49  }
```

Exercise: Shared Memory

- Objectives
- Tasks
- Discussion
- Solutions