CSC552 – Advanced UNIX Programming

Shared Memory and Message Queues

Dr. L. Frye
Kutztown University

Shared Memory

- Data structure – shmid_ds (page 525)
- shmatt_t
- Functions
  - shmget()
  - shmat()
  - shmdt()
  - shmctl()
    - IPC_RMID
    - IPC_SET
    - IPC_STAT

Examples

- sharedmemory/monitorshared.c

- threads/sharedsum
- sharedmemory/sharedsum

Message Queues

- Data structure – msgid_ds (page 535)
  - msgqnum_t
  - msglen_t

- Functions
  - msgget()
  - msgsnd()

Steps

- Allocate a buffer, *mbuf*, which is of type *mymsg_t* and size

  sizeof(mymsg_t) + strlen(mymessage).
- Copy *mymessage* into the *mbuf→mtext* member.
- Set the message type in the *mbuf→mtype* member.
- Send the message.
- Free *mbuf*.

More Functions

- msgrcv()
  - IPC_RMID
  - IPC_SET
  - IPC_STAT
- msgctl()
  - EACCES
  - EINVAL
  - EPERM

Example

- msgqueues/msgqueuelog.c

Producer-Consumer

- Assumptions
  - All messages are same size
  - Messages sent but not received are buffered automatically by OS
- Total of N messages used
  - N slots in semaphore solution
- #define N 100

```
void producer()
{
    int item;
    message m;            /* message buffer */
    while (TRUE)  {               /* loop forever */
        produce_item(&item);  /* generate next item */
        receive(consumer, &m);  /* wait for an empty to arrive */
        build_message(&m, item);   /* construct a msg to send */
        send(consumer, &m);           /* send item to consumer */
    }
}   /* end producer */
```

```
void consumer()
{
     int item, i;
     message m;

    for (i = 0; i < N; i++)
          send(producer, &m);    /* send N empties */
    while (TRUE)  {              /* loop forever */
         receive(producer, &m);/* get message containing item */
         extract_item(&m, &item);  /* extract item from message */
         send(producer, &m);     /* send back empty reply */
         consume_item();         /* do something with the item */
    }
}   /* end consumer */
```