Overview

- typing "`man function`" on the command line will bring up the "man-page" which describes the function
- many of the functions have corresponding UNIX command line programs: make sure you are looking at the man-page for the function
- almost all of the functions return a -1 for failure and 0 for success
- if a function fails it sets the `errno` value with a number corresponding to the type of error – the numbers can be found in the `errno.h` include file

fork()

- `fork()` spawns an identical child process

```
#include <sys/types.h>
#include <unistd.h>

pid_t fork(void);
```

- example:

```
pid_t pid;

pid = fork();
if( pid == 0 ) {
   // only the child should get here
   while( 1 )
      printf( "I am the child\n" );
} // end if // only the parent should get here
```

execl()

- `execl()` replaces a process with an executable piece of code
- `arg0` is traditionally the name of executable
- list of arguments must end with a `NULL`

```
#include <unistd.h>

int execl(const char *path, const char *arg0, ...,
     const char *argn, char * /*NULL*/);
```

- example:

```
pid = fork();
if( pid == 0 )
   execl( "keyboard", "keyboard", 0 );
// run the keyboard executable
```

getpid()

- `getpid()` returns the current process' id

```
#include <sys/types.h>
#include <unistd.h>

pid_t getpid(void);
```

- example:

```
pid_t pid;

pid = getpid();
```

kill()

- `kill()` sends a signal to a process

```
#include <sys/types.h>
#include <signal.h>

int kill(pid_t pid, int sig);
```

- example:

```
pid = getpid();
kill( pid, SIGQUIT );
```

usleep()

- `usleep()` suspends the current process fo r "useconds" micro-seconds

```
#include <unistd.h>

int usleep(unsigned int useconds);
```

- example:

```
printf( "wake me up in a second\n" );
usleep( 1000000 );
printf( "I woke up!\n" );
```

shmget()

- `shmget()` gets the identifier for a block of shared memory registered with value "key"
- "key" is decided by the programmer, "`IPC_PRIVATE`" will return the next available key – this key can't shared
- different values for "`shmflag`" change how the shared memory is retrieved
- "`IPC_CREAT`" is specified to create new memory

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

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

- example:

```
int smem, smem2;

smem = shmget( IPC_PRIVATE, 100, 0 );
smem2 = shmget( 0x30, 100, IPC_CREAT );
```

ftok()

- `ftok()` returns an identifier based on a file name
- allows easy exchange of keys for IPC functions

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

key_t ftok(const char *path, int id);
```

- example:

```
key_t iKey;

iKey = ftok( "/u4/ctrudeau/queues", 3 );
```

shmctl()

- shmctl() controls a block of shared memory
  - "IPC_STAT" get status info of block
  - "IPC_SET" change permissions and ownership
  - "IPC_RMID" remove a block

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int shmctl(int shmid, int cmd, struct shmid_ds *buf);
```

- example:

```
int smem;

smem = shmget( IPC_PRIVATE, 100, 0 );
shmctl( smem, IPC_RMID, NULL );
// deletes sh mem block
```

shmat()

- shmat() associates a local block of memory with a shared block of memory
- provides a memory mapped interface to shared memory

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

void *shmat(int shmid, void *shmaddr, int shmflg);
```

- example:

```
int smem;
void *my_mem;

smem = shmget( IPC_PRIVATE, 100, 0 );
my_mem = shmat( smem, 0, MSG_R | MSG_W );
```

msgget()

- `msgget()` gets a pointer to a message queue
- "key" is the identifier for the queue – see "`ftok()`"
- "msgflg" sets permissions and creation flags
  - "`IPC_CREAT`" create the queue
  - "`IPC_EXCL`" disallow attaching to existing queue

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgget(key_t key, int msgflg);
```

- example:

```
iPerm = MSG_R | MSG_W | MSG_R >> 3 | MSG_W >> 3;
iPerm |= IPC_CREAT | IPC_EXCL;
iMsgId = msgget( iKey, iPerm );
```

msgctl()

- `msgctl()` controls a message queue
  - "`IPC_STAT`" get queue status information
  - "`IPC_SET`" set permissions and ownership
  - "`IPC_RMID`" remove a queue

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgctl(int msqid, int cmd, struct msqid_ds *buf );
```

- example:

```
iRet = msgctl( iMsgId, IPC_RMID, 0 );
```

msgsnd() & msgrcv()

- msgsnd() & msgrcv() send and receive messages
  - "IPC_NOWAIT" don't block for the message
- all message structures passed should begin with an integer which indicates the message type
- the message type field is not included in the size calculation

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgsnd(int msqid, const void *msgp, size_t msgsz,
     int msgflg);

int msgrcv(int msqid, void *msgp, size_t msgsz,
     long msgtyp, int msgflg);
```

Send & Receive Examples

```
typedef struct _m_init
{
   int iType;
   char cEXNum;
} m_init;

m_init pMsg;
```

⋮

```
iRet = msgsnd( iMsgId, pMsg, msgSize - sizeof( int ),
          0 );
iRet = msgsnd( iMsgId, pMsg, msgSize - sizeof( int ),
          IPC_NOWAIT );

iRet = msgrcv( iMsgId, pMsg, msgSize - sizeof( int ),
          msgType, 0 );
```