Today

- Message Passing
- Shared Memory
- Signals
Message Queue

- Create/connect to a message queue
  - `int msgget(key_t key, int msgflg)`
  - `key` is a system-wide unique identifier
  - `msgflg tells msgget() what to do with the queue`
  - `msgid = msgget(key, 0666 | IPC_CREAT)`
Message Queue

- Sending to the queue
  - struct msgbuf {
    long mtype;
    char mtext[100];
  };
  - int msgsnd(int msqid, const void *msgp,
     size_t msgsz, int msgflg);
  - msqid is the id of the queue
  - msgp is a pointer to the message
  - msgsz is the actual size of the message
  - msgflg indicates other options, you can set it to 0
Message Queue

- Receiving from the queue
  - `int msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg)`
  - `msgp` is the buffer to store the message
  - `msgsz` is the size of the message
  - `msgtyp` is used to identify which message to receive
  - `msgflg` indicates other options, you can set it to 0
Message Queue

- Destroying a message queue
  
  ```c
  #include <sys/msg.h>
  
  msgctl(msqid, IPC_RMID, NULL);
  
  You can destroy the IPC by ipcrm command.
  ```
Message Queue Example

- **Sending messages (Sender)**
  - Set up the message queue
  - Read content from the stdin line by line
  - Write them into the message queue

- **Receiving messages (Receiver)**
  - Connect to the message queue
  - Read message from the queue
  - Print them out
Shared Memory

- Creating the segment and connecting
  - int shmget(key_t key, size_t size, int shmflg)
  - shmid = shmget(key, 1024, 0644 | IPC_CREAT);

- Attach – getting a pointer to the segment
  - void* shmat(int shmid, void *shmaddr, int shmflg)
  - data = shmat(shmid, (void *) 0, 0);
Shared Memory

- Reading and writing
  - Returned data is a string, so you can use printf
    `printf("shared contents: %s\n", data);`

- Detaching from and deleting segments
  - Detach – `int shmdt(void *shmaddr)`
  - Destroy – `shmctl(shmid, IPC_RMID, NULL)`
Shared Memory Example

- **shm_server.c**
  - Create a shared memory section
  - Write something into the shared memory
  - Wait until shm_client changes the first char to '*

- **shm_client.c**
  - Attach the shared memory created by shm_server
  - Read the content
  - Change the first character to '*
Signals

• Signals can be sent to a process with the following call: `int kill(int pid, int sig)`.
• Each signal is identified by a number.
• When a signal is received the following can happen:
  1. default action
  2. signal is ignored or blocked
  3. signal is caught and handled
Blocking Signals

- Each process has a signal mask, which indicates which signals are blocked.
- The signal mask can be set with: `int sigprocmask(int how, sigset_t *set, sigset_t *old)`.
- You can specify which signals to block, unblock or just reset the whole mask.
- There are functions that you can use to manipulate the set: `sigemptyset`, `sigaddset`, `sigismember`.
Handling Signals

• The first thing one has to do is define a handler function. This function is executed when the signal arrives:
  
  ```c
  void handler(int signum)
  {
    printf("Signal has arrived!\n")
    ...
  }
  ```

• Once a signal handler function is defined, you can install a signal handler using the `sigaction` call.
sigaction

struct sigaction new_action; //needed struct
new_action.sa_handler = handler;
new_action.sa_flags = 0; //special options
sigemptyset(&new_action.sa_mask);
//signals to block
sigaction(SIGQUIT, &new_action, NULL);
• Alternative:
The sigaction struct also has a field called sa_sigaction. It allows you to specify handle function which takes in more than just the signal number.
• (check out “man sigaction”)
SIGALRM

• SIGALRM is a signal sent to a process once a time limit has elapsed.
• Defined in the signal.h header.
• Can be used to make a longrunning action time out or to perform an action periodically.
• Can be scheduled from alarm(int seconds) function.
• SIGALRM and alarm are used to implement Sleep().
SIGALRM Exercise

• Write a signal handler that handles SIGALRM and prints “Woke up for the # time”

• Call alarm(5) three times.
Questions?