CS 460
Process Control
And Communication

Functions

```
int execl(const char *path, const char *arg, ...)

int execlp(const char *file, const char *arg, ...)

int execle(const char *file, const char *arg, ...,
   char const* envp[])

int execv(const char *path, char *const argv[])

int execvp(const char *file, char *const argv[])

int dup2(int oldfd, int newfd)

int pipe(int filedes[2])

pid_t waitpid(pid_t pid, int *status, int options)

char* strtok_r(char *str, const char* delim, char **saveptr)
```

dup2()

```
#define MAXLEN 1024

/* This code works on Zeus! */
int main()
{
   /* dup2()  makes newfd be the copy of oldfd,
    * closing newfd first if necessary.
    */

   char data[MAXLEN];
   int fd;

   fd = open("test.txt", O_WRONLY | O_CREAT | O_TRUNC, S_IRWXU);
   dup2(fd, STDOUT_FILENO);

   fprintf(stderr,"> ");
   fgets(&(data[0]),MAXLEN,stdin);

   write(fd, &(data[0]), strlen(data));

   printf("%s\n", data);

   close(fd);

   printf("ONCE MORE: %s\n", data);
}
```

simple pipe()

```
#define MAXLEN 1024
#define READ 0
#define WRITE 1

/* This code works on Zeus! */
int main()
{
    char dataPipeWrite[MAXLEN];
    char dataPipeRead[MAXLEN];
    int thePipe[2];
    pid_t childPid;

    memset(&(dataPipeWrite[0]), '\0', MAXLEN);
    memset(&(dataPipeRead[0]), '\0', MAXLEN);

    pipe(thePipe);
                                        /* get data from user */
    readFromCommandLine(dataPipeWrite, MAXLEN);

    write(thePipe[WRITE], &(dataPipeWrite[0]), strlen(dataPipeWrite));
    read(thePipe[READ], &(dataPipeRead[0]), MAXLEN);

    fprintf(stderr,"READ FROM PIPE: %s\n",&(dataPipeRead[0]));

    close(thePipe[WRITE]);
    close(thePipe[READ]);
}
```

pipe()

```
void readFromCommandLine(char * data, int maxSize)
{
    fprintf(stderr,"> ");
    fgets(data, maxSize, stdin);
}

/* This code works on Zeus! */
int main()
{
  /* pipe(int filedes[2])  creates  a  pair of file
   * descriptors, pointing to a pipe inode, and places
   * them in the array pointed to by filedes.
   * filedes[0] is for reading,
   * filedes[1] is for writing.
   */

    char data[MAXLEN];
    int thePipe[2];
    pid_t childPid;

    memset(&(data[0]), '\0', MAXLEN);

    pipe(thePipe);

    childPid = fork();
```

pipe()

```
if(childPid == 0)
{
    /* I AM A CHILD */
    close(thePipe[WRITE]);
    read(thePipe[READ], &(data[0]), MAXLEN);

    while(strncmp( &(data[0]), "STOP", 4) != 0 )
    {
        printf("CHILD> %s\n", &(data[0]));
        read(thePipe[READ], &(data[0]), MAXLEN);
    }
    close(thePipe[READ]);
}
else
{
    close(thePipe[READ]);

    readFromCommandLine(&(data[0]), MAXLEN);
    write(thePipe[WRITE], &(data[0]), strlen(data));

    while(strncmp(&(data[0]), "STOP", 4) != 0)
    {
        readFromCommandLine(&(data[0]), MAXLEN);
        write(thePipe[WRITE], &(data[0]), strlen(data));
    }
    close(thePipe[WRITE]);
}
```