3 Signals

1.
```
typedef void (*sighandler_t) (int);
sighandler_t signal(int signum, sighandler_t handler);
handlers:
SIG_IGN, SIG_DFL, SIG_ERR

int kill(pid_t pid, int sig);
int raise(int signo);
unsigned int alarm(unsigned int seconds);
int pause(void);
```

2.
```
int sigemptyset(sigset_t* set);
int sigfillset(sigset_t* set);
int sigaddset(sigset_t* set, int signum);
int sigdelset(sigset_t* set, int signum);
int sigismember(const sigset_t* set, int signum);
int sigpending(sigset_t *set);
int sigsuspend(const sigset_t *mask);

int sigprocmask(int how, const sigset_t *set, sigset_t *oldset
   );
how:
        SIG_BLOCK
        SIG_UNBLOCK
        SIG_SETMASK

int sigaction(int signum, const struct sigaction *act, struct
   sigaction *oldact);
        struct sigaction{
                void (*sa_handler)(int);
                sigset_t sa_mask;
                int sa_flags;
        };
        flags:
                SA_RESTART

int sigsetjmp(sigjmp_buf env, int savesigs);
void siglongjmp(sigjmp_buf env, int val);
```

3.
```
void abort(void);
int system(const char *command);
unsigned int sleep(unsigned int seconds);
```

4 Inter Process Communication

1.
```
int pipe(int pipefd[2]);

FILE *popen(const char *command, const char *type);
        type: "r" or "w"

int pclose(FILE *stream);
```

2.
```
int mkfifo(const char *pathname, mode_t mode);
        mode: O_NONBLOCK, O_CREAT, O_EXCL
```

3.
```
key_t ftok(const char *pathname, int proj_id);

int msgget(key_t key, int msgflg);
key: IPC_PRIVATE
flag: IPC_CREAT, IPC_EXCL

int msgctl(int msqid, int cmd, struct msqid_ds *buf);
cmd: IPC_STAT, IPC_SET, IPC_RMID
struct msqid_ds{
        struct ipc_perm msg_perm;
        time_t  msg_stime;
        time_t  msg_rtime;
        time_t  msg_ctime;
        msgqnum_t msg_qnum;
        pid_t msg_lspid;
        pid_t msg_lrpid;
};

int msgsnd(int msqid, const void *msgp, size_t msgsz, int
   msgflg);
ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long
   msgtyp, int msgflg);
        for msgp:
        struct msgbuf{
                long mtype;
                char mtext[1];
        };
```

4.
```
int semget(key_t key, int nsems, int semflg);
        key: IPC_PRIVATE
        semflg: IPC_CREAT, IPC_EXCL

int semctl(int semid, int semnum, int cmd, ...);

int semop(int semid, struct sembuf *sops, unsigned nsops);
struct sembuf{
        unsigned short sem_num;
        short sem_op;
        short sem_flg;
}
sem_flg: IPC_NOWAIT, SEM_UNDO
```

5.
```
int shmget(key_t key, size_t size, int shmflg);
shmflg: IPC_CREAT, IPC_EXCL

int shmctl(int shmid, int cmd, struct shmid_ds *buf);
struct shmid_ds{
        struct ipc_perm shm_perm;
        time_t  shm_atime;
        time_t  shm_dtime;
        time_t  shm_ctime;
        pid_t shm_cpid;
        pid_t shm_lpid;
        ...
};

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

Note:

1. This may not be complete.
2. Usually header files that need to be included are:

```
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>
#include <signals.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/msg.h>
#include <sys/sem.h>
```

Other header files may be needed.