System And Devicve Programming
Unix System Calls

Processes

- pid_t getpid();
- pid_t getppid();
- pid_t fork (void);
- pid_t wait (int *statLoc);
    - The status information are collected using macros defined in <sys/wait.h>
    - WIFEXITED, WIFSIGNALED, etc.
- pid_t waitpid (pid_t pid, int *statLoc, int options);
- int execl (char *path, char *arg0, ..., (char *)0);
- int execlp (char *name, char *arg0, ..., (char *)0);
- int execle (char *path, char *arg0, ..., (char *)0, char *envp[]);
- int execv (char *path, char *argv[]);
- int execvp (char *name, char *argv[]);
- int execve (char *path, char *argv[], char *envp[]);
- int system (const char *string);

Signals

- void (*signal (int sig, void (*func)(int)))(int);
    - signal (SIGname, SIG_DFL)
    - signal (SIGname, SIG_IGN)
    - signal (SIGname, signalHandlerFunction)
- int kill (pid_t pid, int sig);
- int raise (int sig);
- int pause (void);
- unsigned int alarm (unsigned int seconds);

Threads

- int pthread_equal (pthread_t tid1, pthread_t tid2);
- pthread_t pthread_self (void);
- int pthread_create (pthread_t *tid, const pthread_attr_t *attr, void *(*startRoutine)(void *), void *arg);
- void pthread_exit (void *valuePtr);
- int pthread_join (pthread_t tid, void **valuePtr);
- int pthread_cancel (pthread_t tid);
- int pthread_detach (pthread_t tid);
    - pthread_attr_setdetachstate (&attr, PTHREAD_CREATE_DETACHED);
    - PTHREAD_CREATE_JOINABLE);

File

- int open (const char *path, int flags, mode_t mode);
- int read (int fd, void *buf, size_t nbytes);
- int write (int fd, void *buf, size_t nbytes);
- off_t lseek (int fd, off_t offset, int whence);

  - whence = SEEK_SET, SEEK_CUR, or SEEK_END
- `int close (int fd);`

Filesystem

- `int stat (const char *path, struct stat *sb);`
- `int lstat (const char *path, struct stat *sb);`
- `int fstat (int fd, struct stat *sb);`
  - `struct stat {`
  - `  mode_t st_mode;     /* file type & mode */`
  - `  ino_t st_ino;       /* i-node number */`
  - `  dev_t st_dev;       /* device number */`
  - `  dev_t st_rdev;      /* device number */`
  - `  ...`
  - `};`
- `char *getcwd (char *buf, int size);`
- `int chdir (char *path);`
- `int mkdir (const char *path, mode_t mode);`
- `int rmdir (const char *path);`
- `DIR *opendir (const char *filename);`
- `struct dirent *readdir (DIR *dp);`
- `int closedir (DIR *dp);`
- struct dirent {

```
  inot_t d_no;
  char d_name[NAM_MAX+1];
  ...
}
```

Non-Blocking I/O

- `fd = open (name, O_WRONLY | ... | O_NONBLOCK );`
- `int fcntl (int fd, int cmd, ... /* int arg */ );`
  - F_GETFL or F_SETFL: Get/set file status flaG
  - F_DUPFD or F_DUPFD_CLOEXEC: Duplicate an existing descriptor
  - F_GETFD or F_SETFD: Get/set file descriptor flags
  - F_GETOWN or F_SETOWN: Get/set asynchronous I/O ownership
  - F_GETLK, F_SETLK, or F_SETLKW: Get/set record locks

File Locking

- `int fcntl(int fd, int cmd, struct flock *flockptr);`
  - F_GETFL or F_SETFL: Get/set file status flaG
  - F_DUPFD or F_DUPFD_CLOEXEC: Duplicate an existing descriptor
  - F_GETFD or F_SETFD: Get/set file descriptor flags
  - F_GETOWN or F_SETOWN: Get/set asynchronous I/O ownership
  - F_GETLK, F_SETLK, or F_SETLKW: Get/set record locks
- struct flock {

```
  short l_type;
  short l_whence;
  off_t l_start;
  off_t l_len;
  pid_t l_pid;
};
```

I/O Multiplexing

- `int select (int maxfdp1, fd_set *restrict readfds, fd_set *restrict writefds, fd_set *restrict exceptfds, struct timeval *restrict tvptr);`
    - o `tvptr == NULL`, wait forever
    - o `tvptr->tv_sec == 0 && tvptr->tv_usec == 0`, do not wait at all
    - o `tvptr->tv_sec != 0 || tvptr->tv_usec != 0`, e wait the specified number of seconds and microseconds
- `void FD_SET (int fd, fd_set *fdset);`
- `void FD_CLR (int fd, fd_set *fdset);`
- `void FD_ZERO (fd_set *fdset);`
- `int FD_ISSET (int fd, fd_set *fdset);`

Asynchronous I/O

-
  ```
  struct aiocb {
    int aio_fildes;
    off_t aio_offset;
    volatile void *aio_buf;
    size_t aio_nbytes;
    int aio_reqprio;
    struct sigevent aio_sigevent;
    int aio_lio_opcode;
  };
  ```

-
  ```
  struct sigevent {
    int sigev_notify;
    int sigev_signo;
    union sigval sigev_value;
    void (*sigev_notify_function)(union sigval);
    pthread_attr_t *sigev_notify_attributes;
  };
  ```
- `int aio_read(struct aiocb *aiocb);`
- `int aio_write(struct aiocb *aiocb);`
- `int aio_fsync (int op, struct aiocb *aiocb);`
- `int aio_suspend(const struct aiocb *const list[], int nent, const struct timespec *timeout);`
- `int aio_cancel (int fd, struct aiocb *aiocb);`

Memory Mapping

- `void *mmap (void *addr,size_t len,int prot,int flag,int fd,off_t off);`
    - o prot = PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE
    - o flag = MAP_FIXED, MAP_SHARED, MAP_PRIVATE
- `void *munmap (void *addr, size_t len);`

Semaphores

- `int sem_init (sem_t *sem, int pshared, unsigned int value);`
- `int sem_wait (sem_t *sem);`
- `int sem_trywait (sem_t *sem);`
- `int sem_post (sem_t *sem);`

- int sem_getvalue (sem_t *sem, int *valP);
- int sem_destroy (sem_t *sem);

Mutexes

- int pthread_mutex_init (pthread_mutex_t *mutex, const pthread_mutexattr_t *attr);
    - PTHREAD_MUTEX_INITIALIZER
- int pthread_mutex_lock (pthread_mutex_t *mutex);
- int pthread_mutex_trylock (pthread_mutex_t *mutex);
- int pthread_mutex_unlock (pthread_mutex_t *mutex);
- int pthread_mutex_destroy (pthread_mutex_t *mutex);

Reader-Writer Locks

- int pthread_rwlock_init (pthread_rwlock_t *restrict rwlock, const pthread_rwlockattr_t *restrict attr);
- int pthread_rwlock_rdlock (pthread_rwlock_t *rwlock);
- int pthread_rwlock_wrlock (pthread_rwlock_t *rwlock);
- int pthread_rwlock_unlock (pthread_rwlock_t *rwlock);
- int pthread_rwlock_tryrdlock(pthread_rwlock_t *rwlock);
- int pthread_rwlock_trywrlock(pthread_rwlock_t *rwlock);
- int pthread_rwlock_timerdlock(pthread_rwlock_t *restrict rwlock, const struct timespec *restrict tsptr);
- int pthread_rwlock_timedwrlock(pthread_rwlock_t *restrict rwlock, const struct timespec *restrict tsptr);
- int pthread_rwlock_destroy (pthread_rwlock_t *rwlock);

Condition Variables

- int pthread_cond_init (pthread_cond_t *restrict cond, const pthread_condattr_t *restrict attr);
- int pthread_cond_wait (pthread_cond_t *restrict cond, pthread_mutex_t *restrict mutex);
- int pthread_cond_timedwait (pthread_cond_t *restrict cond, pthread_mutex_t *restrict mutex, const struct timespec *restrict tsptr);
- int pthread_cond_signal (pthread_cond_t *cond);
- int pthread_cond_broadcast (pthread_cond_t *cond);
- int pthread_cond_destroy (pthread_cond_t *cond);

Spin Locks

- int pthread_spin_init (pthread_spinlock_t *lock, int pshared);
- int pthread_spin_lock (pthread_spinlock_t *lock);
- int pthread_spin_trylock (pthread_spinlock_t *lock);
- int pthread_spin_unlock (pthread_spinlock_t *lock);
- int pthread_spin_destroy (pthread_spinlock_t *lock);

Barriers

- int pthread_barrier_init (pthread_barrier_t *restrict barrier, const pthread_barrierattr_t *restrict attr, unsigned int count);
- int pthread_barrier_wait (pthread_barrier_t *barrier);

- int pthread_barrier_destroy (pthread_barrier_t *barrier);

Pipes

- int pipe (int file_descr[2]);

FIFOs

- int mkfifo (const char *path, mode_t mode);
- int mkfifoat (int fd, const char *path, mode_t mode);

Keys

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

Message Queues

- int msgget (key_t key, int flag);
- int msgctl (int msqid, int cmd, struct msqid_ds *buf);
  - cmd = IPC_STAT, IPC_SET, IPC_RMID
- int msgsnd (int msqid, const void *ptr, size_t nbytes, int flag);
- ssize_t msgrcv (int msqid, void *ptr, size_t nbytes, long type, int flag);

Shared Memory

- int shmget (key_t key, size_t size, int flag);
- int shmctl (int shmid, int cmd, struct shmid_ds *buf);
- void *shmat (int shmid, const void *addr, int flag);
- int shmdt (const void *addr);