isec Engenharia

Sistemas Operativos

Engenharia Informática - DEIS

2021/22

API Unix — Ficha de apoio a exame (parte prática apenas)

Processos e programas — Processes and programs

```
pid_t fork(void)

pid_t wait(int *status)
pid_t waitpid(pid_t pid, int *status, int options)
  -- Algumas Macros para status:
        WIFEXITED, WEXITSTATUS, WIFSIGNALED, WTERMSIG

int execl(const char *path, const char *arg, ...)
int execlp(const char *file, const char *arg, ...)
int execle(const char *path, const char *arg, ...,
                                     char * const envp[]);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
```

Sinais — Signals

```
typedef void (*sighandler_t)(int);

sighandler_t signal(int signum, sighandler_t handler)
int kill(pid_t pid, int sig);

int sigaction(int signum, const struct sigaction *act,
                                     struct sigaction *oldact)
   -- Alguns campos em sigaction:
          void (*sa_handler)(int)
          void (*sa_sigaction)(int, siginfo_t *, void *)

int sigqueue(pid_t pid, int sig, union sigval value)

int alarm(int nseconds);
```

Threads POSIX — Threads POSIX

```
   pthread_attr_t
   pthread_t
int pthread_create(pthread_t * thread,
                   const pthread_attr_t  * attr,
                   void *(*start_routine) (void *),
                   void * arg)
thread_t pthread_self(void)
void pthread_exit(void * retval)
int pthread_cancel(pthread_t thread)
int pthread_setcancelstate(int state,int *oldstate)
     -- PTHREAD_CANCEL_ENABLE, PTHREAD_CANCEL_DISABLE
int pthread_setcanceltype(int type, int *oldtype)
     -- PTHREAD_CANCEL_DEFERRED, PTHREAD_CANCEL_ASYNCHRONOUS
int pthread_join(pthread_t thread, void ** retval)
int pthread_kill(pthread_t thread, int sig)
```

Mutexes (pthread) — Mutexes (pthread)

```
   pthread_mutex_t

int pthread_mutex_init(pthread_mutex_t *mutex,
            const pthread_mutexattr_t *mutexattr)
int pthread_mutex_lock(pthread_mutex_t *mutex)
int pthread_mutex_unlock(pthread_mutex_t *mutex)
int pthread_mutex_destroy(pthread_mutex_t *mutex)
```

Variáveis condicionais — Conditional Variables

```
   pthread_cond_t MyCondVar;

pthread_cond_init (pthread_cond_t *)
pthread_cond_t MyCondVar = PTHREAD_COND_INITIALIZER;
pthread_cond_destroy (pthread_cond_t *)
```

```
pthread_cond_wait (pthread_cond_t *,pthread_mutex_t *)

pthread_cond_timedwait (pthread_cond_t *,
                        pthread_mutex_t *, const struct timespec *)

pthread_cond_signal (pthread_cond_t * )

pthread_cond_broadcast(pthread_cond_t *);
```

Semáforos — Semaphores

```
int semget(key_t key, int nsems, int semflg)
          -- flags: IPC_PRIVATE IPC_CREAT IPC_EXCL
int semctl(int semid, int semnum, int cmd, ...)
          -- comandos: IPC_RMID, SETVAL
int semop(int semid, struct sembuf *sops, unsigned nsops)
  -- sembuf: unsigned short sem_num  -- núm do sem
                      short  sem_op   -- operação
                      short  sem_flg  -- flags:
                                      -- IPC_NOWAIT, IPC_UNDO
```

Memória partilhada — Shared Memory

```
int shmget(key_t key, size_t size, int shmflg)
     -- flags: IPC_PRIVATE IPC_CREAT IPC_EXCL

void *shmat(int shmid, const void *shmaddr, int shmflg)
     -- flags: SHM_RDONLY

int shmdt(const void *shmaddr)

int shmctl(int shmid, int cmd, struct shmid_ds *buf)
     -- Alguns comandos: IPC_RMID
```

Ficheiros/pipes — Files / pipes

```
int open(const char *pathname, int flags)
int open(const char *pathname, int flags, mode_t mode)
    -- flags: O_RDONLY, O_WRONLY, O_RDWR, O_CREAT
    -- mode: permissões

int close(int fd)
size_t read(int fd, void *buf, size_t count)
size_t write(int fd,const void *buf,size_t count)


int fcntl(int fd, int cmd)
int fcntl(int fd, int cmd, long arg)
    -- Alguns comandos: F_GETFL, F_SETFL

int dup(int oldfd)
int dup2(int oldfd, int newfd)

int mkfifo(const char *pathname, mode_t mode)
    -- mode: permissões

int unlink(const char *pathname)
```

Select — Select

```
   fd_set
   struct timeval

int select(int nfds, fd_set * readfds,
           fd_set * writefds, fd_set * exceptfds,
           struct timeval * timeout)

void FD_CLR(int fd, fd_set * set)
int FD_ISSET(int fd, fd_set * set)
void FD_SET(int fd, fd_set * set)
void FD_ZERO(fd_set * set)
```