Informática Industrial - Formulario de llamadas POSIX

```
#define _POSIX_C_SOURCE       199309L
#include <unistd.h>
long sysconf(int option);

#include <stdio.h>

FILE *fopen(const char *nombre, const char *modo);
int fclose(FILE *arch);
int fflush(FILE *arch);
int fgetc(FILE *arch); int fputc(int c, FILE *arch);
int getchar(void); int putchar(int car);
char *fgets(char *s, int n, FILE *arch); char *gets(char *s);
int fprintf(FILE *arch, const char *formato, ...);
int sprintf(char *str, const char *formato, ...);
int fscanf(FILE *arch, const char *formato, ...);
int sscanf(char *str, const char *formato, ...);
size_t fread(void *ptr, size_t tam, size_t nobj, FILE *arch);
size_t fwrite(const void *ptr, size_t tam, size_t nobj, FILE *arch);
int fseek(FILE *arch, long offset, int origen); long ftell(FILE *fp);
int feof(FILE *arch); int ferror(FILE *arch);

modo: “r”, “w”, “a”, “r+”, “w+”, “a+”, “rb”, “wb”...
origen: SEEK_SET, SEEK_CUR, SEEK_END
#include <string.h>

char *strcpy(char *dest, const char *orig);
char *strcat(char *dest, const char *orig);
int strcmp(char *cad1, char *cad2);
size_t strlen(char *cad);
void *memset(void *var, int c,  size_t n);

#include <stdlib.h>

void *malloc(size_t tam); void free(void *p);-

#include <sys/types.h>
#include <sys/wait.h> /* wait */

pid_t fork(void);
pid_t getpid(void); pid_t getppid(void);
int execl(const char *ejecutable, const char *arg0, ...., NULL);
void exit(int status);
pid_t wait(int *estado); pid_t waitpid(pid_t, int *estado, int options);
WIFEXITED(estado); WIFSIGNALED(estado); WEXITSTATUS(estado);
```

- options: WNOHANG, WUNTRACED.

```
#include <signal.h>

int sigemptyset(sigset_t *pset);
int sigfillset(sigset_t *pset);
int sigaddset(sigset_t *pset, int sig);
int sigdelset(sigset_t *pset, int sig);
int sigprocmask(int how, const sigset_t *set, sigset *oset);
int sigaction(int sig, struct sigaction *act, struct sigaction *oact);
int kill(pid_t pid, int sig);
int sigqueue(pid_t pid, int sig, const union sigval val);
unsigned int sleep(unsigned int seg);
int sigsuspend(const sigset_t *nueva_mascara);
int sigwaitinfo(const sigset_t *estas_sg, siginfo_t *infop);
int sigtimedwait(const sigset_t *estas_sg, siginfo_t *infop,
                 const struct timespec *timeout);

struct sigaction
{
    void(* sa_handler) ();
    void (* sa_sigaction) (int numsen,
                siginfo_t *datos, void *extra);
    sigset_t sa_mask;
    int sa_flags;
};

union sigval
{
    int sival_int;
    void *sival_ptr;
};

typedef struct
{
    int si_signo;
    int si_code;
    union sigval si_value;
} siginfo_t;

struct sigevent
{
    int sigev_notify;
    int sigev_signo;
    union sigval sigev_value;
    ...
};
```

- Señales: SIGFPE, SIGKILL, SIGTERM, SIGINT, SIGQUIT, SIGUSR1, SIGUSR2, SIGALRM.
- Señales t.r.: SIGRTMIN a SIGRTMAX; RTSIG_MAX.
- how: SIG_BLOCK, SIG_UNBLOCK, SIG_SETMASK
- sa_handler: SIG_DFL, SIG_IGN, puntero a función.
- sa_flags: SA_SIGINFO
- si_code: SI_QUEUE, SI_TIMER, SI_ASYNCIO, SI_MESGQ, SI_USER
- sigev_notify: SIGEV_SIGNAL
- errno: EINTR, EAGAIN

Informática Industrial - Formulario de llamadas POSIX

```
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>

int mkfifo(const char *nombre, mode_t modo);
int open(const char *path, int oflag, mode_t modo);
ssize_t read(int fd, void *buf, size_t nbyte);
ssize_t write(int fd, const void *buf, size_t nbyte);
int close(int descriptor);
```

- Modo: S_I + (R, W, X) + (USR, GRP, OTH). También S_IRWXU, S_IRWXG, S_IRWXO
- Flags: O_RDONLY, O_WRONLY, O_RDWR, O_CREAT, O_EXCL, O_APPEND, O_TRUNC, O_NONBLOCK

```
#include <time.h>

time_t time(time_t *tiempo);
time_t mktime(struct tm *fecha);
int clock_gettime(clockid_t reloj, struct timespec *tiempo);
int clock_getres(clockid_t reloj, struct timespec *resol);
int clock_settime(clockid_t reloj, struct timespec *tiempo);
int nanosleep(struct timespec *retraso, struct timespec *queda);
int timer_create(clockid_t reloj, struct sigevent *aviso, timer_t *tempo);
int timer_settime(timer_t tempo, int flags, const struct itimerspec *spec,
                                            struct itimerspec *spec_ant);
int timer_delete(timer_t tempo);
int timer_gettime(timer_t tempo, struct itimerspec *queda);
int timer_getoverrun(timer_t tempo);

struct timespec {
    time_t tv_sec;
    long tv_nsec;
};

struct itimerspec {
    struct timespec it_value;
    struct timespec it_interval;
};

struct tm {
    int tm_hour;    /* 0 - 23 */
    int tm_min;     /* 0 - 59 */
    int tm_sec;     /* 0 - 59 */
    int tm_mon;     /* 0 - 11 */
    int tm_mday;    /* 1 - 31 */
    int tm_year;    /* 0 es 1900 */
};
```

Constantes: CLOCK_REALTIME, DELAYTIMER_MAX.

```
#include <sys/socket.h>
#include <netinet/if.h>
#include <arpa/inet.h>

int socket(int dominio, int tipo, int protocolo);
int bind(int socket, const struct sockaddr *direccion, socklen_t long_dir);
int listen(int socket, int tam_cola);
int accept(int socket, struct sockaddr *direccion, socklen_t *lon_dir);
int connect(int socket, const struct sockaddr *direccion, socklen_t tam);
ssize_t send(int id_socket, const void *pdatos, size_t tam, int flagss);
ssize_t sendto(int id_socket, const void *pdatos, size_t tam, int flagss,
                    const struct sockaddr *destino, socklen_t tam_dest);
ssize_t recv(int id_socket, void *buffer, size_t tam, int flagsr);
ssize_t recvfrom(int id_socket, void *buffer, size_t tam, int flagsr,
                        struct sockaddr *origen, socklen_t *tam_origen);
int shutdown(int id_socket, int opcion);
int getsockname(int id_socket, struct sockaddr *direccion, socklen_t *tam_dir);
int getpeername(int id_socket, struct sockaddr *direccion, socklen_t *tam_dir);
int getsockopt(int id_socket, int nivel, int param, const void *val, socklen_t *tam);
int setsockopt(int id_socket, int nivel, int param, const void *val, socklen_t tam);
uint32_t htonl(uint32_t local); uint16_t htons(uint16_t local);
uint32_t ntohl(uint32_t dered); uint16_t ntohs(uint16_t dered);
in_addr_t inet_addr(const char *dirc);
char *inet_ntoa(struct in_addr dir);

struct sockaddr {
    sa_family_t sa_family;
    char sa_data[];
}

struct sockaddr_in {
    sa_family_t    sin_family;
    in_port_t      sin_port;
    struct in_addr sin_addr;
}

struct in_addr {
    in_addr_t s_addr;
    (…)
}
```

- tipo: SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET
- dominio: AF_UNIX, AF_INET, AF_INET6
- protocolo: IPPROTO_TCP, IPPROTO_UDP
- s_addr: INADDR_ANY
- flagsr: MSG_OOB, MSG_EOR; MSG_NOSIGNAL
- flagss: MSG_OOB, MSG_PEEK, MSG_WAITALL
- opcion: SHUT_RD, SHUT_RW, SHUT_RDWR
- nivel: SOL_SOCKET, IPPROTO_TCP

param: SO_ACCEPTCON, SO_BROADCAST, SO_DEBUG, SO_DONTROUTE, SO_ERROR, SO_KEEPALIVE, SO_LINGER, SO_OOBINLINE, SO_RCVBUF, SO_RCVLOWAT, SO_TYPE, SO_RCVTIMEO, SO_REUSEADDR, SO_SNDBUF, SO_SNDLOWAT, SO_SNDTIMEO

Informática Industrial - Formulario de llamadas POSIX

```
#define _POSIX_C_SOURCE 199506L
#include <pthread.h>

int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                                   void *(* rut_com)(void *), void *arg);
void pthread_exit(void *valor);
int pthread_join(pthread_t thread, void **valor);
int pthread_cancel(pthread_t thread);
int pthread_detach(pthread_t thread);
int pthread_equal(pthread_t t1, pthread_t t2);
pthread_t pthread_self(void);

int pthread_sigmask(int how, const sigset_t *set, sigset_t *oset);
int pthread_kill(pthread_t thread, int sig);
int pthread_setcancelstate(int state, int *oldstate);
int pthread_setcanceltype(int type, int *oldtype);
void pthread_testcancel(void);
```

- state: PTHREAD_CANCEL_ENABLE, PTHREAD_CANCEL_DISABLE
- type: PTHREAD_CANCEL_DEFERRED, PTHREAD_CANCEL_ASYNCRONOUS

```
int pthread_attr_init(pthread_attr_t *attr);
int pthread_attr_destroy(pthread_attr_t *attr);

int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);
int pthread_attr_setinheritsched(pthread_attr_t *attr, int inherit);
int pthread_attr_setschedparam(pthread_attr_t *attr, const struct sched_param *param);
int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);
int pthread_attr_setscope(pthread_attr_t *attr, int scope);
int pthread_attr_setstackaddr(pthread_attr_t *attr, void *stackaddr);
int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize);

int pthread_attr_getdetachstate(pthread_attr_t *attr, int *detachstate);
int pthread_attr_getinheritsched(pthread_attr_t *attr, int *inherit);
int pthread_attr_getschedparam(pthread_attr_t *attr, const struct sched_param *param);
int pthread_attr_getschedpolicy(pthread_attr_t *attr, int *policy);
int pthread_attr_getscope(pthread_attr_t *attr, *int scope);
int pthread_attr_getstackaddr(pthread_attr_t *attr, void *stackaddr);
int pthread_attr_getstacksize(pthread_attr_t *attr, size_t *stacksize);
```

- detachstate: PTHREAD_CREATE_JOINABLE, PTHREAD_CREATE_DETACHED
- inherit: PTHREAD_INHERIT_SCHED, PTHREAD_EXPLICIT_SCHED
- policy: SCHED_FIFO, SCHED_RR, SCHED_OTHER
- scope: PTHREAD_SCOPE_ SYSTEM, PTHREAD_SCOPE_PROCESS
- stacksize: PTHREAD_STACK_MIN

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

int pthread_mutexattr_init(pthread_mutexattr_t *attr);
int pthread_mutexattr_destroy(pthread_mutexattr_t *attr);
int pthread_mutexattr_getpshared(pthread_mutexattr_t *attr, int *pshared);
int pthread_mutexattr_getprioceiling(pthread_mutexattr_t *attr, int *prioceiling);
int pthread_mutexattr_getprotocol(pthread_mutexattr_t *attr, int *protocol);
int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr, int pshared);
int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr, int prioceiling);
int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr, int protocol);
```

- Inicializador: PTHREAD_MUTEX_INITIALIZER
- protocol:PTHREAD_PRIO_INHERIT, PTHREAD_PRIO_PROTECT,
  PTHREAD_PRIO_NONE
- pshared: PTHREAD_PROCESS_SHARED, PTHREAD_PROCESS_PRIVATE

```
int pthread_cond_init(pthread_cond_t *cond, pthread_condattr_t *attr);
int pthread_cond_broadcast(pthread_cond_t *cond);
int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_timedwait(pthread_cond_t *cond,
                           pthread_mutex_t *mutex, const struct timespec *abstime);
int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex);
int pthread_cond_destroy(pthread_cond_t *cond);
int pthread_condattr_init(pthread_condattr_t *attr);
int pthread_condattr_getshared(pthread_condattr_t *attr, int *pshared);
int pthread_condattr_setpshared(pthread_condattr_t *attr, int pshared);
```

- Inicializador: PTHREAD_COND_INITIALIZER
- process_shared: PTHREAD_PROCESS_SHARED, PTHREAD_PROCESS_PRIVATE

Informática Industrial - Formulario de llamadas POSIX

```
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <limits.h>

int shm_open(const char *nombre, int oflag, mode_t modo);
int ftruncate(int fd, off_t tamano);
int shm_unlink(const char *nombre);
void *mmap(void *donde, size_t long, int protec, int mapflags, int fd, off_t offset);
int munmap(void *comienzo, size_t long);
```

- oflag y modo: Como **open**.
- protec: PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE
- mapflags: MAP_SHARED, MAP_PRIVATE, MAP_FIXED
- constantes: PAGESIZE, _SC_PAGESIZE

```
#include <semaphore.h>
#include <sys/stat.h>
#include <fcntl.h>
sem_t *sem_open(const char *nombre, int oflags, mode_t modo, unsigned int vinic);
int sem_close(sem_t *sema);
int sem_unlink(const char *nombre);
int sem_wait(sem_t *sema);
int sem_trywait(sem_t *sema);
int sem_post(sem_t *sema);
int sem_getvalue(sem_t *sema, int *valor);
int sem_init(sem_t *donde, int compartido, unsigned int vinic);
int sem_destroy(sem_t *donde);
```

- flags, modo: Como **open**

```
#include <mqueue.h>

mqd_t mq_open(const char *mq_name, int oflag, mode_t modo, struct mq_attr *atributos);
int mq_send(mqd_t cola, const char *datos, size_t longitud, unsigned int prioridad);
int mq_receive(mqd_t cola, const char *datos, size_t longitud, unsigned int *prioridad);
int mq_getattr(mqd_t cola, struct mq_attr *atributos);
int mq_close(mqd_t cola);
int mq_unlink(const char *nombre);
int mq_notify(mqd_t cola, const struct sigevent *espec);

struct mq_attr {
                long mq_maxmsg;
                long mq_msgsize;
                long mq_flags;
                long mq_curmsgs;
                };
```

Constante: MQ_PRIO_MAX

- oflag y modo: Como **open**.