ENTRADA / SALIDA CON FICHEROS: LLAMADAS A SISTEMA

Cabecera	Sintaxis de la función
sys/stat.h fcntl.h	int **creat**(char* nombre, mode_t modo); int **open**(char* nombre, int flags [, mode_t modo]);
unistd.h	int **access**(const char *nombre, int modo); int **close**(int fd); int **dup**(int fd); int **dup2**(int oldfd, int newfd); int **lseek**(int fd, int offset, int whence); int **read**(int fd, void *buffer, int count); int **write**(int fd, const void *buffer, int count);

ENTRADA / SALIDA CON FICHEROS: FUNCIONES DE BIBLIOTECA

Cabecera	Sintaxis de la función
stdio.h	void **clearerr**(FILE * f); int **fclose**(FILE *f); FILE *__fdopen__(int fd, const char *modo); int **feof**(FILE *f); int **ferror**(FILE *f); int **fflush**(FILE *f); int **fgetc**(FILE *f); char ***fgets**(char s, int s, FILE *f); int **fileno**(FILE *f); FILE ***fopen**(const char *nombre, const char *modo) int **fprintf**(FILE *f, const char *formato, ...); int **fputc**(int c, FILE *f); int **fputs**(const char *cad, FILE *f) size_t **fread**(void *b, size_t t, size_t n, FILE *f); FILE *__freopen__(const char *nombre, const char *modo, FILE *f); int **fscanf**(FILE *f, const char *formato, ...); int **fseek**(FILE *f, long offset, int whence); long int **ftell**(FILE *f); size_t **fwrite**(const void *p, size_t i1, size_t i2, FILE *f); int **getc**(FILE *f); int **putc**(int c, FILE *f); int **remove**(const char *nombre); int **rename**(const char *viejo, const char *nuevo); FILE *__tmpfile__(void); int **ungetc**(char c, FILE *f);

ENTRADA / SALIDA ESTÁNDAR, CONVERSIÓN Y COMPROBACIÓN DE FORMATO

Cabecera	Sintaxis de la función
stdio.h	int **getchar**(void); char *__gets__(char *cad); int **printf**(const char *formato, ...); int **putchar**(int c); int **puts**(const char *cadena) int **scanf**(const char *formato, ...); int **perror**(char *cadena); int **sprintf**(char *cad, const char *formato, ...); int **sscanf**(const char *cad, const char *formato, ...);
stdlib.h	double **atof**(const char *nptr); int **atoi**(const char *cadena)
ctype.h	int **isalnum** (int c); Funciones similares: **isalpha / isascii / isblank / iscntrl / isdigit / isgraph / islower / isprint / ispunct / isspace / isupper / isxdigit** int **tolower**(int c); int **toupper**(int c);

MANIPULACIÓN DE STRINGS

Cabecera	Sintaxis de la función

string.h	void *__memchr__(const void *s, int c, size_t n);
	int **memcmp**(const void *s1, const void *s2, size_t n);
	void *__memcpy__(void *s1, const void *s2, size_t n);
	void *__memmove__(void *s1, const void *s2, size_t n);
	void *__memset__(void *s, int c, size_t n);
	char *__strcat__(char*s1, const char *s2);
	char *__strchr__(char *s, int c);
	int **strcmp**(const char*s1, const char *s2);
	char *__strcpy__(char s1, const char s2);
	char *__strdup__(const char *s);
	char *__strerror__(int errnum);
	size_t **strlen**(const char *s);
	char *__strncat__(char*s1, const char *s2, size_t n);
	int **strncmp**(const char *s1, const char *s2, size_t n);
	char *__strncpy__(char *s1, const char *s2, size_t n);
	char *__strpbrk__(const char *s1, const char *s2);
	char *__strrchr__(char *s, int c);
	char *__strstr__(char *s1, const char *s2);
	char *__strtok__(char *s1, const char *s2);

MANIPULACIÓN DE PROCESOS Y PIDS

Cabecera	Sintaxis de la función
unistd.h	int **execl**(const char *path, const char *arg, ...); int **execle**(const char *path, const char *arg, ..., char * const envp[]); int **execlp**(const char *file, const char *arg, ...); int **execv**(const char *path, char *const argv[]); int **execve**(const char *filename, char *const argv[], char *const envp[]); int **execvp**(const char *file, char *const argv[]); int **execvpe**(const char *file, char *const argv[], char *const envp[]); pid_t **fork** (void);
stdlib.h	void **exit**(int status);
sys/types.h unistd.h	pid_t **getpid**(void); pid_t **getppid**(void);
sys/wait.h	pid_t **wait**(int *status); pid_t **waitpid**(pid_t pid, int *status, int options);

CREACIÓN DE TUBERÍAS

Cabecera	Sintaxis de la función
unistd.h	int **pipe**(int filedes[2]); int **open**(char* fifo, int flags); int **mkfifo**(char *fifo, mode_t mode); int **unlink** (char *fifo);

MEMORIA DINÁMICA

Cabecera	Sintaxis de la función

stdlib.h	void *__calloc__(size_t num_datos, size_t tam_dato);
	void **free**(void *dir_memoria);
	void *__malloc__(size_t num_bytes);
	void *__realloc__(void *ptr, size_t tam_bytes);

unistd.h	int **brk**(void *ptr); void *__sbrk__(int incremento);

MANIPULACIÓN DE DIRECTORIOS

Cabecera	Sintaxis de la función
unistd.h	int **chdir**(char* name); char* **getcwd** (char* buf, size_t size); int **rmdir**(const char *pathname);
sys/types.h sys/stat.h	int **mkdir** (char* nombre, mode_t modo);

MANEJO DE SEÑALES

Cabecera	Sintaxis de la función
unistd.h	unsigned int **alarm**(unsigned int seconds); int **pause**(void);
sys/types.h signal.h	int **kill**(pid_t pid, int sig);
signal.h	int **raise**(int señal); void (***signal**(int señal, void (*func)(int))(int); int **sigaction**(int sig, struct sigaction *act, struct sigaction *oldact); int **sigaddset**(sigset_t *set, int sig); int **sigdelset**(sigset_t *set, int sig); int **sigemptyset**(sigset_t *set); int **sigfillset**(sigset_t *set); int **sigismember**(sigset_t *set, int sig); Tipo: **struct sigaction { void (*sa_handler)(); sigset_s sa_mask; int sa_flags; };** Señales: **SIGABRT**, **SIGALRM**, **SIGCHLD**, **SIGCONT**, **SIGFPE**, **SIGHUP**, **SIGILL**, **SIGINT**, **SIGKILL**, **SIGPIPE**, **SIGQUIT**, **SIGSEGV**, **SIGTERM**, **SIGUSR1**, **SIGUSR2**, **SIGSTOP**, etc.

THREADS

Cabecera	Sintaxis de la función
pthread.h	int **pthread_attr_destroy**(pthread_attr_t *attr); int **pthread_attr_init**(pthread_attr_t *attr); int **pthread_create**(pthread_t *thread, pthread_attr_r *attr, void *(*funcion) (void *), void *arg); pthread_t **pthread_self**(void); int **pthread_join**(pthread thid, int *res); int **pthread_exit**(void *value);

MUTEX

Cabecera	Sintaxis de la función
pthread.h	int **pthread_mutex_destroy** (pthread_mutex_t *mutex); int **pthread_mutex_init** (pthread_mutex_t *mutex, pthread_mutexattr_t *attr); int **pthread_mutex_lock** (pthread_mutex_t *mutex); int **pthread_mutex_trylock** (pthread_mutex_t *mutex); int **pthread_mutex_unlock** (pthread_mutex_t *mutex);

MUTEX Y CONDICIONES

Cabecera	Sintaxis de la función
pthread.h	int **pthread_cond_broadcast**(pthread_cond_t *cond); int **pthread_cond_destroy**(pthread_cond_t *cond); int **pthread_cond_init**(pthread_cond_t *cond, pthread_condattr_t *attr); int **pthread_cond_signal**(pthread_cond_t *cond); int **pthread_cond_wait**(pthread_cond_t *cond, pthread_mutex_t_ *mutex);

MANEJO DE SEMÁFOROS

Cabecera	Sintaxis de la función
semaphore.h	int **sem_destroy**(sem_t *sem); int **sem_getvalue**(sem_t *sem, int *val); int **sem_init**(sem_t *sem, int pshared, unsigned int val); int **sem_post**(sem_t *sem); int **sem_unlink**(const char *name); int **sem_wait**(sem_t *sem);
semaphore.h fcntl.h	sem_t ***sem_open**(const char *name, int flags, mode_t mode, unsigned int value)

VARIOS

Cabecera	Sintaxis de la función
stdlib.h	char ***getenv**(const char *name); int **rand**(void); void **srand**(unsigned u);
unistd.h	unsigned int **sleep**(unsigned int seconds);

Control de formato (`printf`, `scanf`, etc)

Código	Formato
%c	Carácter
%d, %i	Entero
%f, %g	Float o double
%p	Puntero
%s	String
%x, %X	Hexadecimal

Whence en `lseek` y `fseek`

Código	Origen
SEEK_SET	Desde el comienzo
SEEK_CUR	Desde la posición actual
SEEK_END	Desde el final

Modo en `fopen`, `fdopen` y `freopen`

Código: modo de apertura y posición del cursor	
"r"	Sólo lectura, al principio
"w"	Sólo escritura, al principio, con truncamiento
"a"	Escritura por el final (concatenación)
"r+"	Lectura/escritura, al principio
"w+"	lectura/escritura, al principio, con truncamiento
"a+"	Lectura al principio y escritura por el final (concatenación)

Modo en `open`, `creat` y `sem_open`

Permisos	Significado
S_IRUSR, S_IWUSR, S_IXUSR	RWX usuario
S_IRGRP, S_IWGRP, S_IXGRP	RWX grupo
S_IROTH, S_IWOTH, S_IXOTH	RWX resto

Flags en `open`, `sem_open`, etc

Flag	Significado
O_RDONLY	Sólo lectura
O_WRONLY	Sólo escritura
O_RDWR	Lectura/escritura
O_APPEND	Concatenación
O_CREAT	Se crea fichero si no existe
O_TRUNC	Trunca el fichero

Modo en `access`

Código	Origen
F_OK	El fichero existe
R_OK	Permiso de lectura
W_OK	Permiso de escritura
X_OK	Permiso de ejecución