Formato de Función	Argumentos	Funcionalidad (explicación breve)	Resultado Devuelto
int execve(const char *pathname, char *const argv[], char *const envp[]);	Path, argv, envp	Ejecuta un programa, reemplazando el proceso actual con el nuevo.	Int (status)
int dup(int oldfd); int dup2(int oldfd, int newfd);	Old fd, new fd	Duplica un descriptor de archivo, opcionalmente a un descriptor específico.	Int (nuevo descriptor)
int pipe(int filedes[2]);	int filedes[2]	Crea un par de descriptores de archivo apuntando a un pipe.	Int (estado)
char *strerror(int errnum);	int errnum	Retorna un mensaje de error asociado con el número de error.	char* (mensaje)
const char *gai_strerror(int ecode);	int ecode	Retorna una descripción legible del error para errores de getaddrinfo.	const char* (mensaje)
extern int errno;	-	Variable global que contiene el último código de error.	Int (código error)
pid_t fork(void);	-	Crea un proceso hijo que es copia del padre.	Int (PID del hijo)
int socketpair(int domain, int type, int protocol, int sv[2]);	Domain, type, protocol, int sv[2]	Crea un par de sockets conectados entre sí.	Int (estado)
uint16_t htons(uint16_t hostshort); uint32_t htonl(uint32_t hostlong);	Number	Convierte valores entre el host y el orden de bytes de red (short y long).	UInt (valor convertido)
uint16_t ntohs(uint16_t netshort); uint32_t ntohl(uint32_t netlong);	Number	Convierte valores entre el orden de bytes de red y el host (short y long).	UInt (valor convertido)
int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);	maxfd, readfds, writefds, exceptfds, timeout	Monitorea múltiples file descriptors.	Int (número de fds listos)
int poll(struct pollfd *fds, nfds_t nfds, int timeout);	fds, nfds, timeout	Espera eventos en múltiples file descriptors.	Int (número de fds)
int epoll_create(int size); **int epoll_ctl(int epfd, int op, int fd, struct epoll_event *event);** **int epoll_wait(int epfd, struct epoll_event *events, int maxevents, int timeout);**	Various	Proporciona una interfaz para I/O event notification.	Int (varía)
int kqueue(void); **int kevent(int kq, const struct kevent *changelist, int nchanges, struct kevent *eventlist, int nevents, const struct timespec *timeout);**	Various	Proporciona una interfaz para notificaciones de eventos en BSD.	Int (varía)

Formato de Función	Argumentos	Funcionalidad (explicación breve)	Resultado Devuelto
int socket(int domain, int type, int protocol);	Domain, type, protocol	Crea un endpoint para comunicación.	Int (descriptor)
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);	sockfd, addr, addrlen	Acepta una conexión en un socket.	Int (descriptor)
int listen(int sockfd, int backlog);	sockfd, backlog	Marca un socket para escuchar conexiones entrantes.	Int (estado)
ssize_t send(int sockfd, const void *buf, size_t len, int flags); **ssize_t recv(int sockfd, void *buf, size_t len, int flags);**	sockfd, buf, len, flags	Envia y recibe datos a través de un socket conectado.	ssize_t (bytes enviados/recibidos)
int chdir(const char *path);	Path	Cambia el directorio de trabajo actual.	Int (estado)
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);	sockfd, addr, addrlen	Asocia una dirección a un socket.	Int (estado)
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);	sockfd, addr, addrlen	Establece una conexión a un socket.	Int (estado)
int getaddrinfo(const char *node, const char *service, const struct addrinfo *hints, struct addrinfo **res);	Node, service, hints, res	Proporciona una interfaz de red independiente para resolver nombres.	Int (estado)
void freeaddrinfo(struct addrinfo *res);	struct addrinfo* res	Libera la memoria dinámica asignada por getaddrinfo.	void
int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen);	sockfd, level, optname, optval, optlen	Configura opciones en sockets.	Int (estado)
int getsockname(int sockfd, struct sockaddr *addr, socklen_t *addrlen);	sockfd, addr, addrlen	Obtiene la dirección local asociada con un socket.	Int (estado)
struct protoent *getprotobyname(const char *name);	Name	Obtiene información del protocolo correspondiente al nombre dado.	struct protoent*
int fcntl(int fd, int cmd, ... /* arg */);	fd, cmd, ...	Manipula el descriptor de archivo.	Int (varía)
int close(int fd);	fd	Cierra un descriptor de archivo.	Int (estado)
ssize_t read(int fd, void *buf, size_t count); **ssize_t write(int fd, const void *buf, size_t count);**	fd, buf, count	Lee y escribe en un descriptor de archivo.	ssize_t (bytes leídos/escritos)

Formato de Función	Argumentos	Funcionalidad (explicación breve)	Resultado Devuelto
pid_t waitpid(pid_t pid, int *status, int options);	pid, status, options	Espera cambios en el estado de un proceso hijo.	Int (PID del proceso)
int kill(pid_t pid, int sig);	pid, sig	Envía una señal a un proceso.	Int (estado)
sighandler_t signal(int signum, sighandler_t handler);	int signum, sighandler_t handler	Establece un manejador de señales.	Sighandler_t (anterior manejador)
int access(const char *pathname, int mode);	Path, mode	Comprueba los permisos de acceso de un archivo.	Int (estado)
int stat(const char *pathname, struct stat *statbuf);	Path, struct stat* buf	Obtiene información sobre el archivo indicado.	Int (estado)
int open(const char *pathname, int flags, mode_t mode);	Pathname, flags, mode	Abre un archivo.	Int (descriptor)
DIR *opendir(const char *name);	Name	Abre un directorio.	DIR* (puntero a directorio)
struct dirent *readdir(DIR *dirp);	DIR* dirp	Lee la siguiente entrada en un directorio.	struct dirent*
int closedir(DIR *dirp);	DIR* dirp	Cierra un directorio abierto.	Int (estado)