ΚΛΗΣΕΙΣ ΣΥΣΤΗΜΑΤΟΣ

ΚΛΗΣΗ	ΠΕΡΙΓΡΑΦΗ
int **open**(const char *path, int oflag, ...);	Άνοιγμα ή δημιουργία αρχείου για διάβασμα ή γράψιμο.
ssize_t **read**(int fd, void *buffer, size_t n);	Ανάγνωση n bytes από περιγραφέα fd στο buffer
ssize_t **write**(int fd, void *buffer, size_t n);	Εγγραφή n bytes από buffer σε περιγραφέα fd
off_t **lseek**(int fd, off_t offset, int whence);	Τοποθέτηση θέσης ανάγνωσης/εγγραφής σε offset bytes από τη θέση του whence. Το whence είναι SEEK_SET ή SEEK_CUR ή SEEK_END.
int **ftruncate**(int fd, off_t length);	Κόψιμο ή επέκταση αρχείου σε τελικό μέγεθος ίσο με length bytes
int **fsync**(int fd);	Σύγχρονη αποθήκευση δεδομένων στον δίσκο
int **close**(int fd);	Κλείσιμο περιγραφέα αρχείου
int **unlink**(const char *path);	Καταργεί τον σύνδεσμο στο αρχείο
int **dup**(int fd); int **dup2**(int fd, int fd2);	Δημιουργία αντιγράφου ενός περιγραφέα αρχείου. Η dup2(fd, fd2) εκτελεί τη λειτουργία fd2 = dup(fd);
pid_t **fork**();	Δημιουργία νέας διεργασίας. Επιστρέφει την τιμή 0 στο παιδί ή το process id του παιδιού στον γονιό.
pid_t **getpid**();	Επιστρέφει το process id της διεργασίας.
pid_t **getppid**();	Επιστρέφει το process id του γονιού της διεργασίας.
void **_exit**(int status); void **exit**(int status);	Τερματίζει τη διεργασία κι επιστρέφει κωδικό status. Η exit εκτελεί επιπλέον διαδικασίες καθαρισμού.
pid_t **wait**(int *stat); pid_t **waitpid**(pid_t pid, int *stat, int options);	Αναμονή για τερματισμό ή αλλαγή κατάστασης διεργασίας-παιδιού.
int **execlp**(const char *file, const char *arg0, ...);	Εκτέλεση προγράμματος file με ορίσματα arg0,...
int **execvp**(const char *file, char *const argv[]);	Εκτέλεση προγράμματος file με διάνυσμα ορισμάτων argv
int **pipe**(int fd[2]);	Δημιουργία ανώνυμου αγωγού
int **mkfifo**(const char *path, mode_t mode);	Δημιουργία επώνυμου αγωγού
int **fcntl**(int fd, int cmd, ...);	Έλεγχος λειτουργίας περιγραφέα fd
int **sigemptyset**(sigset_t *set);	Αρχικοποίηση άδειου συνόλου σημάτων
int **sigfillset**(sigset_t *set);	Αρχικοποίηση γεμάτου συνόλου σημάτων
int **sigaddset**(sigset_t *set, int signo);	Προσθήκη σήματος signo στο σύνολο set
int **sigdelset**(sigset_t *set, int signo);	Αφαίρεση σήματος signo από το σύνολο set
int **sigismember**(const sigset_t *set, int signo);	Ελέγχει αν το σήμα signo ανήκει στο σύνολο set
int **sigprocmask**(int how, const sigset_t *set, sigset_t *oset);	Καθορισμός μάσκας μπλοκαρίσματος. Το how μπορεί να είναι SIG_BLOCK, SIG_UNBLOCK ή SIG_SETMASK
int **sigpending**(sigset_t *set);	Αποθηκεύει εκκρεμή σήματα στο set, τα οποία μπορούν να ελεγθούν με χρήση sigismember.
int **pause**(); int **sleep**(int secs);	Αναμονή παραλαβής/χειρισμού σήματος

int **sigaction**(int sig, const struct sigaction *act, struct sigaction *oact);	Θέτει τον επιθυμητό χειρισμό του σήματος sig. Τα "χρήσιμα" πεδία του struct sigaction είναι: sa_handler που προσδιορίζει τον επιθυμητό χειρισμό (SIG_DFL ή SIG_IGN ή συνάρτηση), sa_flags που προσδιορίζει flags όπως το SA_RESTART.
volatile sig_atomic_t	Τύπος καθολικής μεταβλητής που μπορεί να χρησιμοποιηθεί σε χειριστή σήματος.
int **sigwait**(const sigset_t *set, int *sig);	Αναμονή μέχρι κάποιο από τα σήματα του set να εκκρεμεί
int **sigsuspend**(const sigset_t *sigmask);	Αλλαγή μάσκας μπλοκαρίσματος και αναμονή σήματος
int **kill**(pid_t pid, int sig);	Αποστολή σήματος sig στη διεργασία pid

int **setitimer**(int which, const struct itimerval *restrict nval, struct itimerval *restrict oval);	Ρυθμίζει χρονόμετρο (μέσω της παραμέτρου nval) για να στέλνει αυτόματα SIGALRM στη διεργασία.
	Για μέτρημα πραγματικού χρόνου, which = ITIMER_REAL
	Το struct itimerval έχει πεδία:
	struct timeval it_value; // για το αρχικό "χτύπημα"
	struct timeval it_interval; // για περιοδικό χρονόμετρο
	Το struct timeval έχει πεδία:
	tv_sec (δευτερόλεπτα) , tv_usec (μικροδευτερόλεπτα)

unsigned int **alarm**(unsinged int secs);	Απλό ξυπνητήρι που χτυπά μετά από secs δευτερόλεπτα.

ΒΙΒΛΙΟΘΗΚΗ STDIO

FILE ***fopen**(const char **filename*, const char * *mode*);	Άνοιγμα ή δημιουργία αρχείου για διάβασμα ή γράψιμο.
int **setvbuf**(FILE * stream, char * buffer, int type, size_t size);	Καθορισμός πολιτικής αποθήκης σε τύπο _IOFBF (fully buffered), _IOLBF (line buffered) ή _IONBF (not buffered)
int **fscanf**(FILE *stream, const char *format, ...)	Μορφοποιημένη ανάγνωση από τη ροή stream
int **fprintf**(FILE *stream, const char *format, ...)	Μορφοποιημένη εγγραφή στη ροή stream
size_t **fread**(void *ptr, size_t size, size_t n, FILE *stream)	Ανάγνωση δεδομένων από τη ροή stream
size_t **fwrite**(void *ptr, size_t size, size_t n, FILE *stream)	Εγγραφή δεδομένων στη ροή stream
char ***fgets**(char *restrict s, int n, FILE * stream);	Ανάγνωση μιας γραμμής (ή μέχρι n-1 χαρακτήρες). Το '\n' αποθηκεύεται στο s.
ssize_t **getline**(char **line, size_t *n, FILE *stream);	Ανάγνωση μιας γραμμής, δεσμεύοντας μνήμη για το line.
int **fflush**(FILE *stream);	Αδειασμα αποθήκης δεδομένων (εξόδου μόνο)
int **fclose**(FILE *stream);	Κλείσιμο ροής αρχείου

ΒΙΒΛΙΟΘΗΚΗ STDLIB

void free(void *ptr);	Αποδέσμευση μνήμης στην οποία δείχνει ο ptr.
void *malloc(size_t size);	Δέσμευση μνήμης μεγέθους size bytes
void *calloc (size_t n, size_t size);	Δέσμευση και μηδενισμός μνήμης για πίνακα n αντικειμένων όπου το καθένα έχει μέγεθος size bytes.
void *realloc(void *ptr, size_t size);	Μεταβολή μεγέθους δεσμευμένης μνήμης σε size bytes.