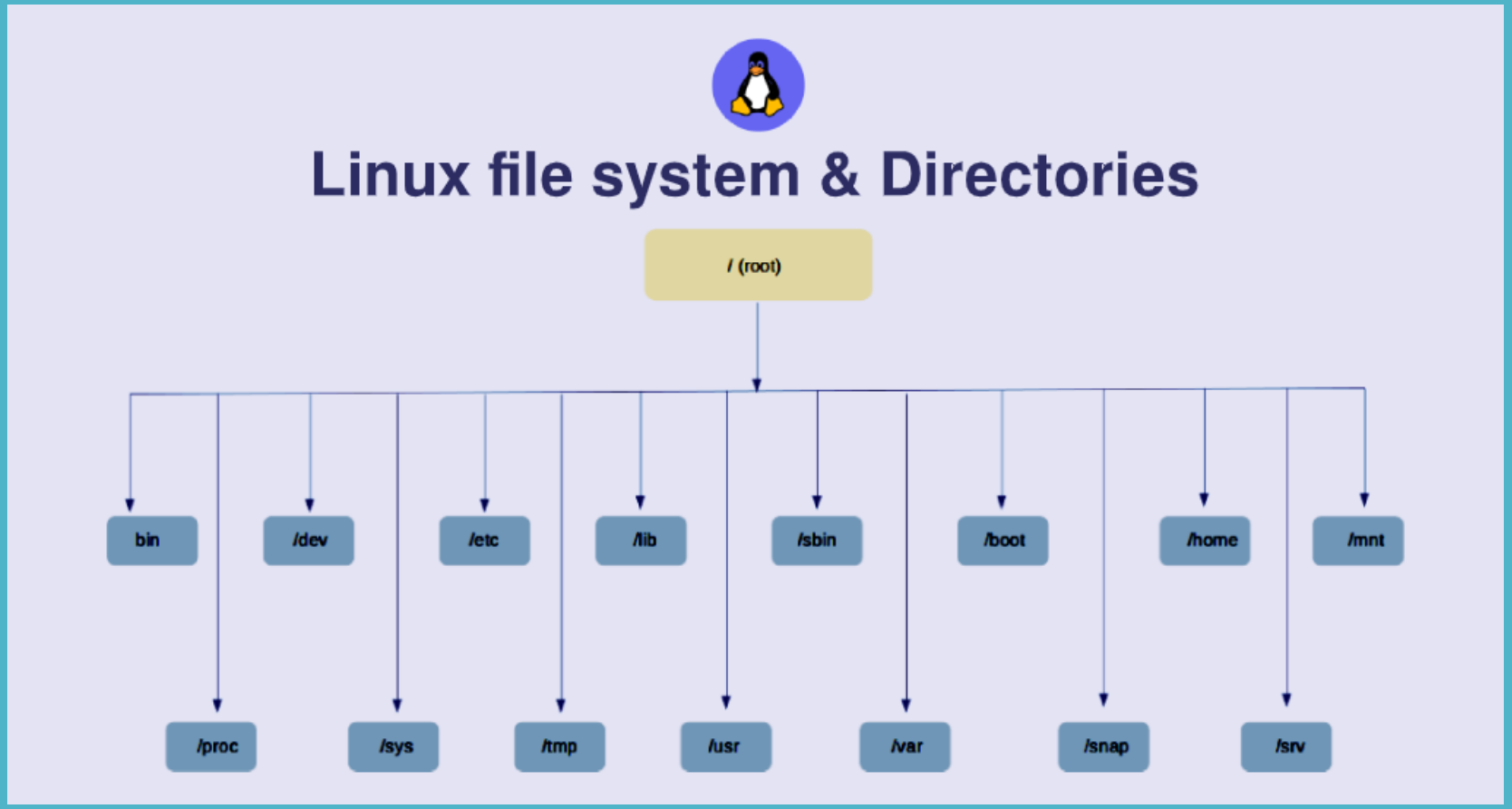
Αρχεία και κατάλογοι

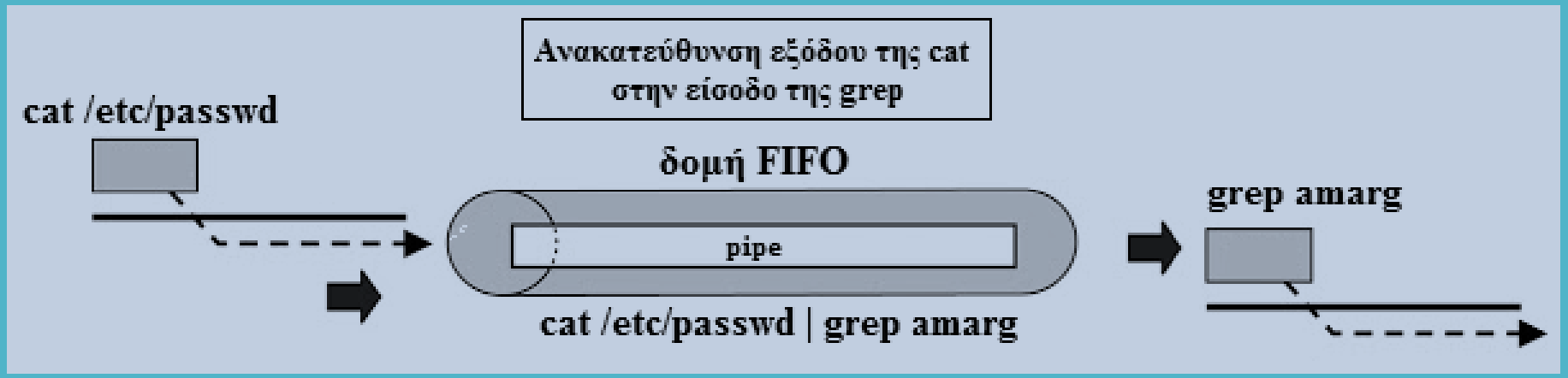


Αρχεία και κατάλογοι

Τα αρχεία αποτελούν μία από τις θεμελιώδεις δομικές μονάδες του λειτουργικού συστήματος Linux, αφού σχεδόν τα πάντα σε αυτό το λειτουργικό σύστημα, περιγράφονται από κατάλληλα αρχεία. Ειδικότερα, στο Linux υφίστανται οι επόμενοι τύποι αρχείων:

1) Συνήθη αρχεία (regular files)→ πρόκειται για τα γνωστά αρχεία των χρηστών που αποτελούν ομάδες bytes αποθηκευμένες στο δίσκο.

2) Αγωγοί ή σωληνώσεις (pipes) → πρόκειται για ειδικά αρχεία τα οποία μπορεί να είναι αποθηκευμένα στο δίσκο του συστήματος αλλά μπορεί και όχι, τα οποία επιτρέπουν την ενδοεπικοινωνία ανάμεσα στις διεργασίες του συστήματος (Interprocess Communication, IPC).



Αρχεία και κατάλογοι

3) Κατάλογοι (directories) → Πρόκειται για αρχεία που αποτελούνται από τη λίστα των αρχείων που περιέχονται σε αυτά.

4) Αρχεία συσκευών (device files) → πρόκειται για ειδικά αρχεία που χρησιμοποιούνται για την αναπαράσταση και προσπέλαση των συσκευών του συστήματος.

```
amarg@amarg-vbox:~$ ls /dev
autofs          ecryptfs  log            net       sg0       tty12  tty25  tty38  tty50  tty63      ttyS17  ttyS3   vboxguest  vcsa5  zero
block           fb0       loop0          null      sg1       tty13  tty26  tty39  tty51  tty7       ttyS18  ttyS30  vboxuser   vcsa6  zfs
bsg             fd        loop1          nvram     shm       tty14  tty27  tty4   tty52  tty8       ttyS19  ttyS31  vcs        vcsu
btrfs-control   full      loop2          port      snapshot  tty15  tty28  tty40  tty53  tty9       ttyS2   ttyS4   vcs1       vcsu1
bus             fuse      loop3          ppp       snd       tty16  tty29  tty41  tty54  ttyprintk  ttyS20  ttyS5   vcs2       vcsu2
cdrom           hidraw0   loop4          psaux     sr0       tty17  tty3   tty42  tty55  ttyS0      ttyS21  ttyS6   vcs3       vcsu3
char            hpet      loop5          ptmx      stderr    tty18  tty30  tty43  tty56  ttyS1      ttyS22  ttyS7   vcs4       vcsu4
console         hugepages loop6          pts       stdin     tty19  tty31  tty44  tty57  ttyS10     ttyS23  ttyS8   vcs5       vcsu5
core            hwrng     loop7          random    stdout    tty2   tty32  tty45  tty58  ttyS11     ttyS24  ttyS9   vcs6       vcsu6
cpu_dma_latency i2c-0     loop-control   rfkill    tty       tty20  tty33  tty46  tty59  ttyS12     ttyS25  udmabuf vcsa       vfio
cuse            initctl   mapper         rtc       tty0      tty21  tty34  tty47  tty6   ttyS13     ttyS26  uhid    vcsa1      vga_arbiter
disk            input     mcelog         rtc0      tty1      tty22  tty35  tty48  tty60  ttyS14     ttyS27  uinput  vcsa2      vhci
dri             kmsg      mem            sda       tty10     tty23  tty36  tty49  tty61  ttyS15     ttyS28  urandom vcsa3      vhost-net
dvd             lightnvm  mqueue         sda1      tty11     tty24  tty37  tty5   tty62  ttyS16     ttyS29  userio  vcsa4      vhost-vsock
```

5) Συμβολικοί σύνδεσμοι (symbolic links) → πρόκειται για αρχεία που περιέχουν τη διαδρομή προς ένα άλλο αρχείο του συστήματος (αντιστοιχούν στα shortcuts των Microsoft Windows).

```
amarg@amarg-vbox:/$ ls -lR / | grep '^l'
lrwxrwxrwx   1 root root          7 Σεπ  19 19:50 bin -> usr/bin
lrwxrwxrwx   1 root root          7 Σεπ  19 19:50 lib -> usr/lib
```

6) Υποδοχείς (sockets) → πρόκειται για αρχεία που επιτρέπουν την ενδοεπικοινωνία των διεργασιών με πιο αποδοτικό και ευέλικτο τρόπο σε σχέση με τους αγωγούς.

Αρχεία και κατάλογοι

- Το κάθε αρχείο του συστήματος προσδιορίζεται με μοναδικό τρόπο από μία ειδική δομή δεδομένων που λέγεται i-node (index node ή information node) και το οποίο περιέχει όλες τις πληροφορίες του αρχείου (δικαιώματα πρόσβασης, μέγεθος, όνομα, κ.τ.λ.).

- Υπάρχουν δύο τύποι i-node, τα i-node μνήμης που διατηρούνται για κάθε ανοιχτό αρχείο και τα i-nodes δίσκου που διατηρούνται για κάθε αρχείο του δίσκου.

- Αυτά τα δύο είδη i-nodes δεν περιέχουν τις ίδιες πληροφορίες.

  - Όταν ανοίγουμε ένα αρχείο το i-node δίσκου αντιγράφεται σε ένα inode-μνήμης.
  - Όταν αποθηκεύουμε ένα αρχείο το i-μνήμης αντιγράφεται σε ένα inode-δίσκου.

```
amarg@amarg-vbox:/$ ls -li
total 970048
     13 lrwxrwxrwx   1 root root         7 Σεπ  19 19:50 bin -> usr/bin
 786433 drwxr-xr-x   3 root root      4096 Σεπ  26 09:47 boot
 802791 drwxrwxr-x   2 root root      4096 Σεπ  19 19:55 cdrom
      2 drwxr-xr-x  18 root root      4000 Σεπ  28 12:37 dev
 262145 drwxr-xr-x 130 root root     12288 Σεπ  26 09:55 etc
 131073 drwxr-xr-x   3 root root      4096 Σεπ  19 19:57 home
     14 lrwxrwxrwx   1 root root         7 Σεπ  19 19:50 lib -> usr/lib
     15 lrwxrwxrwx   1 root root         9 Σεπ  19 19:50 lib32 -> usr/lib32
     16 lrwxrwxrwx   1 root root         9 Σεπ  19 19:50 lib64 -> usr/lib64
     17 lrwxrwxrwx   1 root root        10 Σεπ  19 19:50 libx32 -> usr/libx32
```

Αρχεία και κατάλογοι

Οι πληροφορίες που είναι αποθηκευμένες σε ένα i-node επιστρέφονται ως τα πεδία της επόμενης δομής.

```
struct stat {
        dev_t st_dev;          ID of device containing the file
        ino_t st_ino;          Serial number for the file.
        mode_t st_mode;        Access mode and file type for the file (see Flags).
        nlink_t st_nlink;      Number of links to the file.
        uid_t st_uid;          User ID of file owner.
        gid_t st_gid;          Group ID of group owner.
        dev_t st_rdev;         Device ID (if the file is a character or block special device).
        off_t st_size;         File size in bytes (if the file is a regular file).
        time_t st_atime;       Time of last access.
        time_t st_mtime;       Time of last data modification.
        time_t st_ctime;       Time of last file status change.
        blksize_t st_blksize;  A file system-specific preferred I/O block size for this object.
        blkcnt_t st_blocks;    Number of blocks allocated for this file.
        mode_t st_attr;        The DOS-style attributes for this file (see Flags).
};
```

Αρχεία και κατάλογοι

mode_t st_mode values for file type

Flag	Meaning	
S_IFMT	Type of file.	Η τιμή στο οκταδικό σύστημα
S_IFBLK	File is a block-device special file.	0060000
S_IFCHR	File is a character-device special file.	0020000
S_IFIFO	File is a FIFO special file.	0010000
S_IFREG	File is a regular file.	0100000
S_IFDIR	File is a directory.	0040000
S_IFSOCK	File is a socket.	0140000

Αρχεία και κατάλογοι

Μακροεντολές για τον έλεγχο της κατάστασης του αρχείου (επιστρέφουν TRUE ή FALSE)

Macro	Meaning
S_ISBLK (m)	File is a block-device special file.
S_ISCHR (m)	File is a character-device special file.
S_ISFIFO (m)	File is a FIFO special file.
S_ISREG (m)	File is a regular file.
S_ISDIR (m)	File is a directory.
S_ISSOCK (m)	File is a socket.

Αρχεία και κατάλογοι

Flag	Meaning	
S_IRWXU	Read, write, execute/search permission for owner.	**USER**
S_IRUSR	Read permission for owner.	r
S_IWUSR	Write permission for owner.	w
S_IXUSR	Execute/search permission for owner.	x
S_IRWXG	Read, write, execute/search for group.	**GROUP**
S_IRGRP	Read permission for group.	r
S_IWGRP	Write permission for group.	w
S_IXGRP	Execute/search permission for group.	x
S_IRWXO	Read, write, execute/search for others.	**OTHERS**
S_IROTH	Read permission for others.	r
S_IWOTH	Write permission for others.	w
S_IXOTH	Execute/search permission for others.	x

mode_t st_mode values for permission

Flag	Meaning	
S_IDPOU	Delete, change permission, and take ownership for owner.	**USER**
S_IDUSR	Delete permission for owner.	
S_IPUSR	Change permission permission for owner.	
S_IOUSR	Take ownership permission for owner.	
S_IDPOG	Delete, change permission, and take ownership for group.	**GROUP**
S_IDGRP	Delete permission for group.	
S_IPGRP	Change permission permission for group.	
S_IOGRP	Take ownership permission for group.	
S_IDPOO	Delete, change permission, and take ownership for others.	**OTHERS**
S_IDOTH	Delete permission for others.	
S_IPOTH	Change permission permission for others.	
S_IOOTH	Take ownership permission for others.	
S_ISUID	Set-user-ID on execution	**setuid bit**
S_ISGID	Set-group-ID on execution	**setgid bit**
S_ISVTX	Restricted-deletion flag for directories	**sticky bit**

Αρχεία και κατάλογοι

Παράδειγμα εμφάνισης πληροφοριών αρχείου

```
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
   struct stat sb;
   if (argc != 2) {
        fprintf(stderr, "Usage: %s <pathname>\n", argv[0]);
        exit(EXIT_FAILURE); }
   if (stat(argv[1], &sb) == -1) {
        perror("stat");
        exit(EXIT_FAILURE); }

   printf("File type:                ");
   switch (sb.st_mode & S_IFMT) {
    case S_IFBLK:  printf("block device\n");            break;
    case S_IFCHR:  printf("character device\n");        break;
    case S_IFDIR:  printf("directory\n");               break;
    case S_IFIFO:  printf("FIFO/pipe\n");               break;
    case S_IFLNK:  printf("symlink\n");                 break;
    case S_IFREG:  printf("regular file\n");            break;
    case S_IFSOCK: printf("socket\n");                  break;
    default:       printf("unknown?\n");                break; }

   printf("I-node number:            %ld\n", (long) sb.st_ino);

   printf("Mode:                     %lo (octal)\n", (unsigned long) sb.st_mode);
   printf("Link count:               %ld\n", (long) sb.st_nlink);
   printf("Ownership:                UID=%ld   GID=%ld\n", (long) sb.st_uid, (long) sb.st_gid);
   printf("Preferred I/O block size: %ld bytes\n", (long) sb.st_blksize);
   printf("File size:                %lld bytes\n", (long long) sb.st_size);
   printf("Blocks allocated:         %lld\n", (long long) sb.st_blocks);
   printf("Last status change:       %s", ctime(&sb.st_ctime));
   printf("Last file access:         %s", ctime(&sb.st_atime));
   printf("Last file modification:   %s", ctime(&sb.st_mtime));
   exit(EXIT_SUCCESS); }
```

Αρχεία και κατάλογοι

Παράδειγμα εμφάνισης πληροφοριών αρχείου

```
amarg@amarg-vbox:~$ ./filestat forkex5
File type:                regular file
I-node number:            274381
Mode:                     100775 (octal)
Link count:               1
Ownership:                UID=1000   GID=1000
Preferred I/O block size: 4096 bytes
File size:                17048 bytes
Blocks allocated:         40
Last status change:       Sat Sep 26 20:49:56 2020
Last file access:         Sat Sep 26 20:49:59 2020
Last file modification:   Sat Sep 26 20:49:56 2020
```

```
amarg@amarg-vbox:~$ ./filestat Music
File type:                directory
I-node number:            935608
Mode:                     40755 (octal)
Link count:               2
Ownership:                UID=1000   GID=1000
Preferred I/O block size: 4096 bytes
File size:                4096 bytes
Blocks allocated:         8
Last status change:       Sat Sep 19 20:09:47 2020
Last file access:         Mon Sep 28 12:37:51 2020
Last file modification:   Sat Sep 19 20:09:47 2020
```

```
amarg@amarg-vbox:~$ ls -l forkex5
-rwxrwxr-x 1 amarg amarg 17048 Σεπ  26 20:49 forkex5
```

```
amarg@amarg-vbox:~$ ls -l | grep Music
drwxr-xr-x  2 amarg amarg     4096 Σεπ  19 20:09 Music
```

```
amarg@amarg-vbox:~$ cat /etc/passwd | grep amarg
amarg:x:1000:1000:Athanasios Margaris,,,:/home/amarg:/bin/bash
```

ΔΙΑΠΙΣΤΩΝΟΥΜΕ ΟΤΙ

1) Η κατάσταση (mode) αποτελείται από: (α) τύπο αρχείου, (β) τροποποιητής, (γ) δικαιώματα πρόσβασης
2) Το όνομα του αρχείου ΔΕΝ περιλαμβάνεται στις πληροφορίες του i-node (struct stat).

Mode → 10 0 775
10 → regular file
0 → τροποποιητής (sgt)
775 → r w x r w x r – x

Mode → 04 0 755
04 → directory
0 → τροποποιητής (sgt)
755 → r w x r – x r – x

Επομένως η εντολή ls –l στην πραγματικότητα εκτυπώνει τις τιμές των πεδίων της δομής stat.

Αρχεία και κατάλογοι

Περιγραφέας αρχείου (File Descriptor, fd)

Οι περιγραφείς αρχείου είναι θετικοί ακέραιοι αριθμοί με μικρές τιμές οι οποίοι για κάθε ανοικτό αρχείο υποδεικνύουν τη θέση του στον πίνακα των ανοικτών αρχείων που διατηρεί η κάθε διεργασία.

Κατά την εκκίνηση κάθε διεργασίας αρχικοποιούνται οι επόμενοι περιγραφείς αρχείου

0 (STDIN_FILENO) → standard input (stdin)
1 (STDOUT_FILENO) → standard output (stdout)
2 (STDERR_FILENO) → standard error (stderr)

Για αρκετές από τις συναρτήσεις διαχείρισης αρχείων υφίστανται δύο εκδοχές στην πρώτη εκ των οποίων το αρχείο καθορίζεται από το όνομά του (char * pathname), ενώ στη δεύτερη από τον περιγραφέα αρχείου του (int fd). Για παράδειγμα, η συνάρτηση stat που επιστρέφει μία δομή stat με τις πληροφορίες του i-node ενός αρχείου εμφανίζεται στις επόμενες δύο εκδοχές.

```
int stat (const char * pathname, struct stat * statbuf);
int stat (int fd, struct stat * statbuf);
```

Αρχεία και κατάλογοι

Δικαιώματα πρόσβασης, κάτοχος και ομάδα αρχείου, time stamps

Αν και στη δομή stat υπάρχουν όλες οι πληροφορίες σχετικά με τα δικαιώματα πρόσβασης για το υπό θεώρηση αρχείο, ωστόσο, η ανάκτησή τους διευκολύνεται σημαντικά από τη συνάρτηση

int access (const char * pathname, int mode) ← <unistd.h>

όπου mode μία ή περισσότερες από τις παρακάτω σταθερές συνδυασμένες με τον τελεστή της λογικής διάζευξης (logical OR)

F_OK → το αρχείο υπάρχει?
R_OK → υφίσταται δικαίωμα ανάγνωσης?
W_OK → υφίσταται δικαίωμα εγγραφής?
X_OK → υφίσταται δικαίωμα εκτέλεσης?

Η συνάρτηση επιστρέφει 0 εάν το δικαίωμα πρόσβασης υφίσταται και τη σταθερά EACCESS στην αντίθετη περίπτωση.

Ορισμός δικαιωμάτων πρόσβασης από τον κάτοχο (owner) του αρχείου ή το διαχειριστή (root)

int chmod (const char * pathname, mode_t mode);
int fchmod (int fd, mode_t mode); **← <sys/stat.h>**

Αρχεία και κατάλογοι

```c
#include <errno.h>
#include <stdio.h>
#include <unistd.h>

int main (int argc, char* argv[]) {
 char* path = argv[1];
 int rval;

 /* Check file existence. */
 rval = access (path, F_OK);
 if (rval == 0)
  printf ("%s exists\n", path);
 else {
  if (errno == ENOENT)
   printf ("%s does not exist\n", path);
  else if (errno == EACCES)
   printf ("%s is not accessible\n", path);
  return 0;  }

 /* Check read access. */
 rval = access (path, R_OK);
 if (rval == 0)
  printf ("%s is readable\n", path);
 else
  printf ("%s is not readable (access denied)\n", path);

 /* Check write access. */
 rval = access (path, W_OK);
 if (rval == 0)
  printf ("%s is writable\n", path);
 else if (errno == EACCES)
  printf ("%s is not writable (access denied)\n", path);
 else if (errno == EROFS)
  printf ("%s is not writable (read-only filesystem)\n", path);
 return 0; }
```

```
amarg@amarg-vbox:~$ chmod 144 file1.doc
amarg@amarg-vbox:~$ chmod 344 file2.doc
amarg@amarg-vbox:~$ chmod 544 file3.doc
amarg@amarg-vbox:~$ chmod 744 file4.doc
amarg@amarg-vbox:~$ ls -l *.doc
---xr--r-- 1 amarg amarg    1521 Σεπ  20 19:58 file1.doc
--wxr--r-- 1 amarg amarg 7959821 Σεπ  20 19:58 file2.doc
-r-xr--r-- 1 amarg amarg    2797 Σεπ  20 19:59 file3.doc
-rwxr--r-- 1 amarg amarg     550 Σεπ  20 19:59 file4.doc
amarg@amarg-vbox:~$ ./acc1 file1.doc
file1.doc exists
file1.doc is not readable (access denied)
file1.doc is not writable (access denied)
amarg@amarg-vbox:~$ ./acc1 file2.doc
file2.doc exists
file2.doc is not readable (access denied)
file2.doc is writable
amarg@amarg-vbox:~$ ./acc1 file3.doc
file3.doc exists
file3.doc is readable
file3.doc is not writable (access denied)
amarg@amarg-vbox:~$ ./acc1 file4.doc
file4.doc exists
file4.doc is readable
file4.doc is writable
amarg@amarg-vbox:~$
```

Παράδειγμα χρήσης της access

Αρχεία και κατάλογοι

```c
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include<errno.h>

int main (int argc, char * argv[]) {
  int fd;
  struct stat info;
  if (argc!=2) {
      printf ("Usage testChmod pathname\n");
      return (-1); }
  fd = access(argv[1], F_OK);
  if(fd == -1){
     printf("Error Number : %d\n", errno);
     perror("Error Description");
     return (-2); }
  else {
    stat(argv[1], &info);
    printf("Original file permissions were: %08o\n",
           info.st_mode);
    if (chmod(argv[1], S_IRWXU|S_IRWXG) != 0)
      perror("chmod() error");
    else {
      stat(argv[1], &info);
      printf("After chmod(), file permissions are: %08o\n",
             info.st_mode); }}
   return (0); }
```

```
amarg@amarg-vbox:~$ ls -l | grep results
-rw-rw-r--  1 amarg amarg     262 Σεπ  22 10:08 results
amarg@amarg-vbox:~$ ./testChmod results
Original file permissions were: 00100664
After chmod(), file permissions are: 00100770
amarg@amarg-vbox:~$ ls -l | grep results
-rwxrwx---  1 amarg amarg     262 Σεπ  22 10:08 results
```

Αρχικά γίνεται έλεγχος ύπαρξης του αρχείου με την access και στη συνέχεια καλείται η chmod για να δώσει δικαιώματα r w x στον κάτοχο (S_IRWXU) και στην ομάδα (S_IRWXG). Παρατηρήστε πως αλλάξει όλη η μάσκα δικαιωμάτων και επειδή δεν έχουν οριστεί δικαιώματα για τους υπόλοιπους χρήστες, αυτά έχουν απενεργοποιηθεί.

Παράδειγμα χρήσης της chmod

Αρχεία και κατάλογοι

Δικαιώματα πρόσβασης, κάτοχος και ομάδα αρχείου, time stamps

Τροποποίηση κατόχου και ομάδας κατόχου αρχείου (<unistd.h>)

int chown (const char * pathname, uid_t owner, gid_t group);
int fchown (int fd, uid_t owner, gid_t group);

Τροποποίηση χρονικών σφραγίδων (st_mtime & st_atime)

Α τρόπος (System V → POSIX)
<utime.h>

```
struct utimbuf {
        time_t actime;
        time_t modtime; };
```

int utime (const char * pathname,
struct utimebuf * buf)

Β Τρόπος (BSD Unix)
<systime/h>

```
struct timeval {
        long tv_sec;
        long tv_usec};
```

int utime (const char * pathname,
struct timeval * tvf);

Προεπιλεγμένη μάσκα δικαιωμάτων (sys/stat.h)

int umask (int newmask);

Αρχεία και κατάλογοι

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main (int argc, char * argv[]) {
    int retVal;
    if (argc !=2) {
        printf ("Usage: testChown pathname\n");
        return (-1); }
    retVal = chown (argv[1], (uid_t)3, (gid_t)3);
    if (retVal==-1) {
        printf("Error Number : %d\n", errno);
        perror("Error Description");
        return (-2); }
    printf ("Assignment to %s of user SYS and group SYS
             has been performed succesfully\n", argv[1]);
    return(0); }
```

```
amarg@amarg-vbox:~$ ls -l | grep months.txt
-rw-rw-r--  1 amarg amarg     87 Σεπ  23 14:44 months.txt
amarg@amarg-vbox:~$ ./testChown months.txt
Error Number : 1
Error Description: Operation not permitted
amarg@amarg-vbox:~$ sudo ./testChown months.txt
Assignment to months.txt of user SYS and group SYS
                    has been performed succesfully
amarg@amarg-vbox:~$ ls -l | grep months.txt
-rw-rw-r--  1 sys   sys       87 Σεπ  23 14:44 months.txt
amarg@amarg-vbox:~$
```

Η επιτυχής κλήση της συνάρτησης απαιτεί δικαιώματα διαχειριστή και για το λόγο αυτό εκτελείται μέσα από την εντολή sudo (switch user, do) η οποίο by default προσφέρει root access. Στο παράδειγμα ο νέος owner είναι ο user sys με user id ίσο με 3.

```
amarg@amarg-vbox:~$ cat /etc/passwd | grep sys
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

Παράδειγμα χρήσης της chown

Αρχεία και κατάλογοι

Οι βασικές μορφές διαχείρισης αρχείων

Άνοιγμα / δημιουργία αρχείου – οι εντολές open και creat ← <unistd.h> & <fcntl.h>

int open (char * pathname, int flags, mode_t mode);

- pathname → το όνομα της διαδρομής προς το αρχείο
- flags → το είδος της πρόσβασης [απαιτείται κάποιο από τα O_RDONLY (read only), O_WRONLY (write only) ή O_RDWR (read/write) σε συνδυασμό με κάποια άλλα προαιρετικά flags όπως π.χ, το O_APPEND (τα δεδομένα γράφονται στο τέλος του αρχείου)]
- mode → εάν στα flags συμπεριληφθούν τα O_CREAT ή O_TMPFILE τότε ορίζεται το mode που μπορεί να πάρει κάποιες από τις τιμές S_IRWXU , S_IRUSR, S_IWUSR, S_IXUSR, S_IRWXG, S_IRGRP, S_IWGRP, S_IXGRP, S_IRWXO, S_IROTH, S_IWOTH, S_IXOTH).

int creat (char * pathname, mode_t mode);

Είναι ισοδύναμη με την κλήση της open με flags O_CREAT | O_WRONLY | O_TRUNC .

Οι συναρτήσεις επιστρέφουν τον file descriptor fd προς το αρχείο που άνοιξη ή δημιουργήθηκε. Άλλες συναρτήσεις που επιστρέφουν file descriptors είναι η pipe (που δημιουργεί αγωγούς) και οι socket, accept και connect (network programming).

Κλείσιμο αρχείου → **int close (int fd)** ← <unistd.h>

Αρχεία και κατάλογοι

Οι βασικές μορφές διαχείρισης αρχείων

Ανάγνωση και εγγραφή σε αρχείο – οι συναρτήσεις read και write ← <unistd.h>

Οι εντολές read και write που επιτρέπουν την ανάγνωση δεδομένων από αρχείο και την εγγραφή δεδομένων σε αυτό (όχι όμως για low level διαδικασίες όπως τα message queues), έχουν τη μορφή

int read (int fd, void * buf, size_t length);
int write (int fd, const void * buf, size_t length)

Αμφότερες οι συναρτήσεις επιστρέφουν το πλήθος των bytes που διαβάστηκαν ή εγγράφηκαν (αντίστοιχα) ή την τιμή -1 για την περίπτωση σφάλματος.

Αναζήτηση σε αρχείο τυχαίας προσπέλασης – η συνάρτηση lseek ← <unistd.h>

Η συνάρτηση fseek επιτρέπει τον ορισμό της τρέχουσας θέσης σε αρχείο τυχαίας προσπέλασης και επιστρέφει τη νέα τρέχουσα θέση – η συνάρτηση ορίζεται ως

int lseek (int fd, off_t offset, int whence)

Η παράμετρος whence παίρνει μία από τις τιμές SEEK_SET (αρχή αρχείου), SEEK_CUR (τρέχουσα θέση), SEEK_END (τέλος αρχείου) και δηλώνει από πιο σημείο θα μετακινηθούμε κατά offset bytes με την τιμή του offset να μπορεί να είναι και αρνητική.

Αρχεία και κατάλογοι

Παράδειγμα 1 → Υπολογισμός μεγέθους αρχείου με την lseek

```c
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <errno.h>

int main(int argc, char * argv[]) {
    int fd;
    off_t filelength;
    if (argc !=2) {
        printf ("Usage: flength pathname\n");
        return (-1); }
    fd = open(argv[1], O_RDONLY );
    if (fd < 0) {
        printf("Error Number : %d\n", errno);
        perror("Error Description"); }
    filelength = lseek (fd, 0, SEEK_END);
    if (filelength < 0) {
        printf("Error Number : %d\n", errno);
        perror("Error Description"); }
    else printf ("Size of file %s is %d bytes\n",
                 argv[1], (int)filelength);
    close (fd);
    return (0); }
```

```
amarg@amarg-vbox:~/shared/Lab5$ ls -l *.o
-rwxrwxrwx 1 root root 4224 Σεπ  29 11:43 filestat.o
-rwxrwxrwx 1 root root 2528 Σεπ  29 16:30 fsize1.o
-rwxrwxrwx 1 root root 2200 Σεπ  29 17:19 fsize2.o
-rwxrwxrwx 1 root root 2632 Σεπ  29 20:50 myCopy.o
-rwxrwxrwx 1 root root 2800 Σεπ  29 11:43 testAccess.o
-rwxrwxrwx 1 root root 2768 Σεπ  29 15:28 testChmod.o
-rwxrwxrwx 1 root root 2272 Σεπ  29 15:43 testChown.o
amarg@amarg-vbox:~/shared/Lab5$ ./fsize1 filestat.o
Size of file filestat.o is 4224 bytes
amarg@amarg-vbox:~/shared/Lab5$ ./fsize1 myCopy.o
Size of file myCopy.o is 2632 bytes
amarg@amarg-vbox:~/shared/Lab5$
```

Η lseek επιστρέφει την απόσταση σε bytes της τρέχουσας θέσης από την αρχή του αρχείου. Εάν ως τρέχουσα θέση οριστεί το τέλος του αρχείου, αυτή η απόσταση είναι προφανώς ίση με το τέλος του αρχείου.

Αρχεία και κατάλογοι

Παράδειγμα 2 → Υπολογισμός μεγέθους αρχείου με τη read

```c
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <errno.h>

#define BUFSIZE 100

int main (int argc, char *argv[]) {
    int fd, nread, total=0;
    char buf[BUFSIZE];
    if (argc !=2) {
    printf ("Usage: flength pathname\n");
    return (-1); }
    fd = open(argv[1], O_RDONLY);
    while (1) {
          nread = read(fd,buf,BUFSIZE-1);
          if (nread == 0){
              break; }
         /* buf[nread] = '\0';*/
          total += nread; }
    printf("%d bytes total\n",total);
    close(fd);
    return 0; }
```

```
amarg@amarg-vbox:~/shared/Lab5$ ls -l *.o
-rwxrwxrwx 1 root root 4224 Σεπ  29 11:43 filestat.o
-rwxrwxrwx 1 root root 2528 Σεπ  29 16:30 fsize1.o
-rwxrwxrwx 1 root root 2200 Σεπ  29 17:19 fsize2.o
-rwxrwxrwx 1 root root 2632 Σεπ  29 20:50 myCopy.o
-rwxrwxrwx 1 root root 2800 Σεπ  29 11:43 testAccess.o
-rwxrwxrwx 1 root root 2768 Σεπ  29 15:28 testChmod.o
-rwxrwxrwx 1 root root 2272 Σεπ  29 15:43 testChown.o
amarg@amarg-vbox:~/shared/Lab5$ ./fsize2 filestat.o
4224 bytes total
amarg@amarg-vbox:~/shared/Lab5$ ./fsize2 myCopy.o
2632 bytes total
amarg@amarg-vbox:~/shared/Lab5$
```

Σε κάθε επανάληψη το πρόγραμμα διαβάζει BUSIZE bytes από το αρχείο και προσθέτει το πλήθος των bytes που διάβασε στη μεταβλητή total.

Η read επιστρέφει το πλήθος των bytes που διάβασε και επομένως όταν επιστρέψει 0 σημαίνει πως έχει φτάσει στο τέλος του αρχείου. Στην περίπτωση αυτή η total περιέχει το μέγεθος του αρχείου.

Αρχεία και κατάλογοι

Παράδειγμα 3 → Αντιγραφή αρχείου με τις read και write

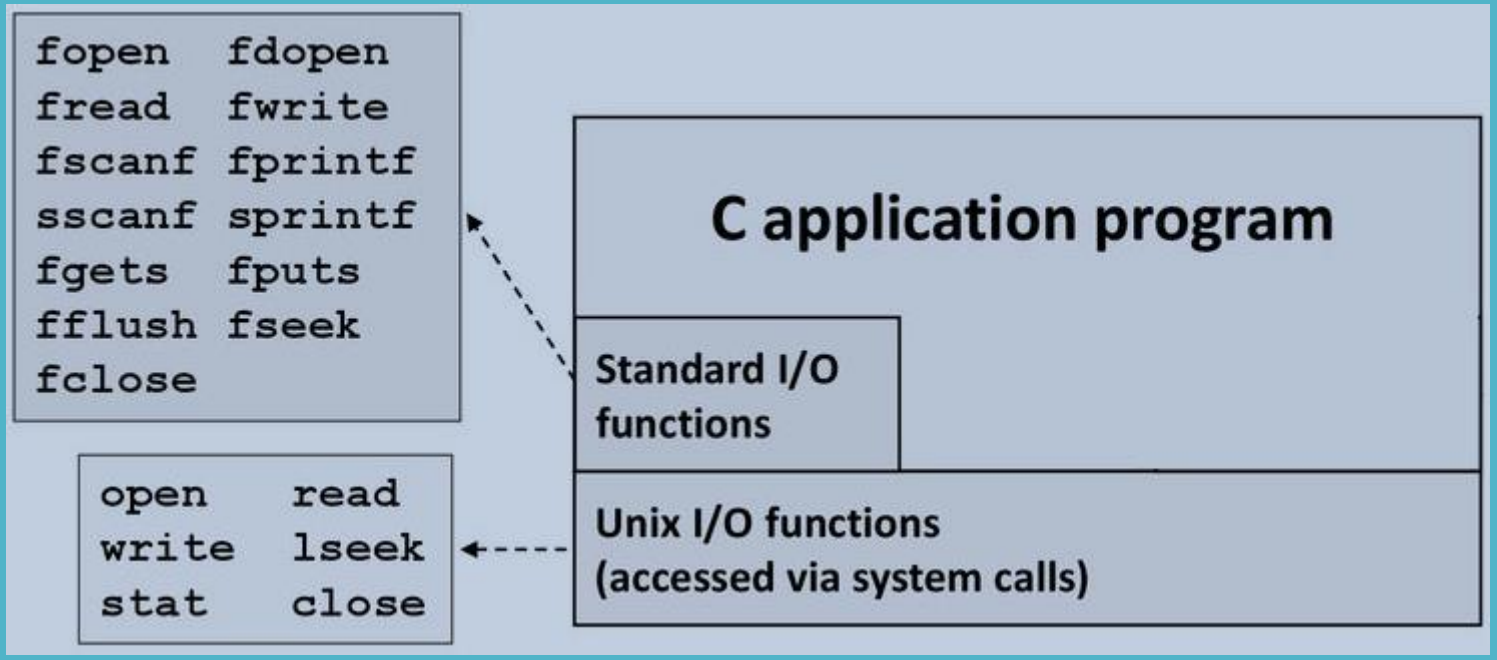
```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#define BUFSIZE 512
```

```c
int main(int argc, char * argv[]) {
    char buffer[BUFSIZE];
    int src, dest, errno;
    int nread, nwritten, n;
    int totalr=0, totalw=0;
    if (argc!=3) {
        printf ("Usage: flength source destination\n");
        return (-1); }
    src = open(argv[1], O_RDONLY);
    if (src < 0) {
        perror("Source file could not be opened!");
        return (-1); }
    dest = open(argv[2], O_CREAT | O_WRONLY | O_TRUNC, S_IRUSR | S_IWUSR);
    if (dest < 0) {
        perror("Target file could not be opened!");
        return (-2); }
    while ((nread = read(src, buffer, BUFSIZE))>0) {
            nwritten=0;
            do {
                n = write (dest, &buffer[nwritten], nread-nwritten);
                nwritten += n;
               } while (nwritten<nread); }
    close (src);
    close (dest);
    return (0); }
```

Για όσο υπάρχουν bytes για ανάγνωση, το πρόγραμμα διαβάζει bytes από το αρχείο εισόδου και τα αποθηκεύει στο αρχείο εξόδου

Αρχεία και κατάλογοι

System I/O vs C Language I/O



SYSTEM I/O → Κλήσεις συστήματος, δεν προχωρά σε ενδιάμεση αποθήκευση και μορφοποίηση της εισόδου

C LANGUAGE I/O → Μέρος της βιβλιοθήκης της C με όνομα libc, πραγματοποιεί μορφοποίηση εισόδου και ενδιάμεση αποθήκευση.

Αρχεία και κατάλογοι

File I/O functions

- `#include <stdio.h>`

function	description
`FILE* fopen(char* filename, char* mode)`	mode is "r", "w", "a"; returns pointer to file or NULL on failure
`int fgetc(FILE* file)` `int fgets(char* buf, int size, FILE* file)`	read a char from a file; read a line from a file
`int fputc(char c, FILE* file)` `int fputs(char* s, FILE* file)`	write a char to a file; write a string to a file
`int feof(FILE* file)`	returns non-zero if at EOF
`int fclose(FILE* file)`	returns 0 on success

`FILE* stdin`
`FILE* stdout`
`FILE* stderr`
	streams representing console input, output, and error

- most return EOF on any error (which is -1, but don't rely on that)

4

```
typedef struct

{

    int level;

    unsigned flags;

    char fd;

    unsigned char hold;

    int bsize;

    unsigned char_FAR* buffer;

    unsigned char_FAR* curp;

    unsigned istemp;

    short token;

}FILE;
```

Αρχεία και κατάλογοι

Παράδειγμα 4 → Αντιγραφή αρχείου με συναρτήσεις της C

```c
int main(int argc, char * argv[]) {
    char cTemp;
    FILE * src, * dest;
    if (argc!=3) {
        printf ("Usage: flength source destination\n");
        return (-1); }

    src = fopen(argv[1], "rb");
    if (!src) {
        printf ("Source File could not be opened! Aborting...");
        return (-2); }

    dest = fopen(argv[2], "wb");
    if (!src) {
        printf ("Destination file could not be opened! Aborting...");
        return (-3); }

    while(fread(&cTemp, 1, 1, src) == 1) {
    fwrite(&cTemp, 1, 1, dest); }

    fclose(src);
    fclose(dest);
    return (0); }
```

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
```

Συναρτήσεις

fopen
fclose
fread
fwrite

Αρχεία και κατάλογοι

Εντολές διαχείρισης καταλόγων

Ανάκτηση καταλόγου εργασίας (unistd.h) → `char * getcwd (char * buf, size_t size)`

Αλλαγή τρέχοντος καταλόγου (unistd.h) →

`int chdir (const char * pathname) & int chdir (int df)`

Αλλαγή ριζικού καταλόγου (unistd.h) → `int chroot (const char * pathname)`

Δημιουργία καταλόγου (unistd.h, fcntl) →

`int mkdir (const char * pathname, mode_t mode)`

Διαγραφή κενού καταλόγου (unistd.h) → `int rmdir (const char * pathname)`

Ανάγνωση περιεχομένων καταλόγου (δομή DIR ← dirent.h)

```
DIR * opendir (const char * pathname)
int closedir (DIR * dir)
struct dirent * readdir (DIR * dir)
```
← **επιστροφή του επόμενου αρχείου στον κατάλογο**

Αρχεία και κατάλογοι

Παράδειγμα 1 → Δημιουργία και διαγραφή καταλόγου

```c
int main(int argc, char * argv[]) {
    long size; int retVal; DIR * dir;
    char * buf, * ptr, arg [20]="ls -l | grep ";
    if (argc!=2) {
        printf ("Usage: dirFun dirname\n");
        return (-1); }
    dir = opendir(argv[1]);
    if (dir) {
        printf ("Directory exists. Aborting...\n");
    closedir(dir);
    return (-2);  }
    size = pathconf(".", _PC_PATH_MAX);
    if ((buf = (char *)malloc((size_t)size)) != NULL)
        ptr = getcwd(buf, (size_t)size);
    printf ("Current Working Directory is %s\n", buf);
    printf ("Creating directory %s inside directory %s\n",
            argv[1], buf);
    free(buf);
    retVal = mkdir (argv[1],0755);
    if (retVal==-1) {
        printf("Error Number : %d\n", errno);
        perror("Error Description");
        return (-2); }
    strcat (arg, argv[1]);
    system (arg);
    printf ("Removing directory %s\n", argv[1]);
    rmdir (argv[1]);
    return (0); }
```

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <dirent.h>
```

```
amarg@amarg-vbox:~$ ./dirFun mydir
Current Working Directory is /home/amarg
Creating directory mydir inside directory /home/amarg
drwxr-xr-x  2 amarg amarg    4096 Σεπ  30 16:03 mydir
Removing directory mydir
```

Επίδειξη των συναρτήσεων mkdir και rmdir

Αρχεία και κατάλογοι

Παράδειγμα 2 → Εκτύπωση περιεχομένων καταλόγου

```c
#include <stdio.h>
#include <sys/types.h>
#include <errno.h>
#include <dirent.h>

int main(int argc, char * argv[]) {
    DIR * dip;
    struct dirent * dit;
    int files = 0;
    if (argc!=2) {
        printf ("Usage: myLs dirname\n");
        return (-1); }
    if ((dip = opendir (argv[1]))==NULL) {
        printf("Error Number : %d\n", errno);
        perror("Error Description");
        return (-2); }
    printf ("List of contents of directory %s\n", argv[1]);
    while ((dit = readdir(dip))!=NULL) {
            files++;
            printf("%s\n", dit->d_name); }
   printf ("Directory %s contains %d file(s)\n", argv[1], files);
   return (0); }
```

```
amarg@amarg-vbox:~$ ./myLs .
List of contents of directory .
.bashrc
months.txt
filestat.o
file2.zip
Templates
forkex5
lex
myCopyc.o
forkex4
pid.c
Downloads
file1.txt
forkex5.c
testChmod
flength
file3.zip
.bash_logout
.vboxclient-draganddrop.pid
.sudo_as_admin_successful
.bash_history
rfile
forkex7
```

Το πρόγραμμα εκτυπώνει τα ονόματα των περιεχομένων του καταλόγου (αρχεία και κατάλογοι), ένα όνομα σε κάθε γραμμή συμπεριλαμβανομένων και των κρυφών αντικειμένων.