Sisteme de operare 1

Curs 6

Titular curs,

Dr. Dragoș Sanda Maria

Apeluri sistem de lucru cu fisiere

- Apeluri sistem Unix
  - => fac apel direct la functiile kernel ale SO.
    (*nivelul inferior de prelucrare a fisierelor*)

- Functiile standard existente in biblioteca C:
  - fopen, fprintf, fscanf, fclose, fseek, etc
    (*nivelul superior de prelucrare a fisierelor*)

Apelul sistem **open**

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

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

- flags=O_RDONLY, O_WRONLY, O_RDWR, O_NDELAY, O_APPEND, O_CREAT, O_EXCL (cu O_CREAT: *creare* ***exclusiva***), O_TRUNC.
- mode= drepturile de acces si functioneaza in concordanta cu *umask*

S_IRUSR, S_IWUSR , S_IXUSR, S_IRGRP, S_IWGRP , S_IXGRP, S_IROTH, S_IWOTH , S_IXOTH

Retur:

-1 –eroare

>0 –descriptorul de fisiere.

man open

Apelul system **creat**

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

int creat(const char * pathname, mode_t mode);

*/*echivalen cu specificarea optiunilor : O_WRONLY|O_CREAT|O_TRUNC la functia **open***/*

man creat

Apelul sistem **close**

#include <unistd.h>

int close (int fd) ;

/*fd – descriptorul de fisier obtinut de **open***/

man close

Functiile **read** si **write**

#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);

ssize_t write(int fd, void *buf, size_t count);

Retur:

0 – EOF

-1 – eroare

>0 –numarul de octeti operati.

man 2 read

man 2 write

Functia **lseek**

#include <sys/types.h>

#include <unistd.h>

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

- offest *: numarul de octeti pe care se face deplasarea*
- whence : *pozitionare relativ la:*
  - SEEK_SET : inceputul fisierului
  - SEEK_CUR: pozitia curenta
  - SEEK_END: sfarsitul fisierului

man lseek

Exemple de apeluri sistem

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

[...]
int fd1, fd2;

fd1 = open ("alina.txt", O_WRONLY | O_TRUNC);
if (fd1 < 0) {
    perror("open" );
    exit( EXIT_FAILURE);
}

fd2 = open ("dan.txt", O_RDWR | O_CREAT, 0644);
...
```

Exemplu complet

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
int main (void)
{
      int fd;
      char *buf;
      ssize_t nocteti;

      /* alocam spatiu pentru buffer−ul de citire*/
      buf = malloc(101);
      if (buf == NULL) {
            perror( "malloc");
            exit(EXIT_FAILURE);
      }

      /* deschidem fisierul */
      fd = open ("gabi.txt", O_RDONLY);
      if (fd < 0) {
            perror("open");
            exit(EXIT_FAILURE);
      }

      /* ne pozitionam în fisier */
      if (lseek (fd, −100, SEEK_END) < 0) {
            perror("lseek");
            exit(EXIT_FAILURE);
      }

      /* citim ultimele 100 caractere in buffer */
      nocteti = read (fd, buf, 100);
      if (nocteti < 0) {
            perror("read");
            exit(EXIT_FAILURE);
      }

      buf [nocteti] = '\0';

      /* afisam sirul citit */
      printf(“ultimii %ld octeti: \n%s\n", nocteti, buf);

      /* inchidem fisierul */
      close (fd);

      /* eliberam buffer−ul alocat */
      free (buf);

      return 0;
}
```

Functiile standard existente in biblioteca C

#include <stdio.h>

FILE ***fopen**(const char *path, const char *mode);

int **fclose**(FILE *fp);

int **fprintf**(FILE *stream, const char *format, ...);

int **fscanf**(FILE *stream, const char *format, ...);

size_t **fread**(void *ptr, size_t size, size_t nmemb, FILE *stream);

size_t **fwrite**(void *ptr, size_t size, size_t nmemb, FILE *stream);

int **fseek**(FILE *stream, long offset, int whence);

man 3 …

Apelurile sistem **dup** si **dup2**

```
#include <unistd.h>

int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

- Duplica descriptorul de fisier *oldfd* astfel:
  - *oldfd* si *newfd* refera acelasi fisier fizic
  - Modul de access se pastreaza de la deschidere
  - Ambii descriptori partajeaza acelasi pointer curent in fisier
- Return:
  -1 – eroare
  >0 - noul descriptor

man dup

Exemplu dup2

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>

main(){
	int oldfd, newfd;

	oldfd = open("ERORI", O_CREAT|O_WRONLY, 0755);
	if(oldfd < 0){
		fprintf(stderr, "pe stderr: open ERORI imposibil\n");
		fprintf(stdout, "pe stdout: open ERORI imposibil\n");
		exit(1);
	}

	newfd = dup2(oldfd, 2);	/*inchide automat fisierul standard stderr*/
				/*noul fisier standard de erori va fi ERORI*/
	if(newfd != 2){
		fprintf(stderr, "pe stderr: dup2 imposibil\n");
		fprintf(stdout, "pe stdout: dup2 imposibil\n");
		exit(1);
	}

	fprintf(stderr, "Mesaj in ERORI prin stderr: dup2 REUSIT\n");
	fprintf(stdout, "Mesaj pe terminal prin stdout: dup2 REUSIT\n");
}
```

Apelul sistem **fcntl**

```
#include <unistd.h>
#include <fcntl.h>

int fcntl(int fd, int cmd, ... /* arg */ );
```

- Furnizeaza sau schimba proprietati ale unui fisier deschis

- cmd:
  - F_DUPFD - Duplicarea unui descriptor
  - F_GETFD sau F_SETFD – flagurile descriptorului de fisiere
  - F_GETFL sau F_SETFL – flagurile de stare a fisierului
  - F_GETOWN sau F_SETOWN – manipularea semnalelor I/O
  - F_GETLK sau F_SETLK – blocarea fisierelor

Blocari de fisiere

- **blocare concilianta** (advisory)
- **blocare obligatorie** (mandatory)
- *blocare exclusiva* - cand un singur proces are access la fisier sau la portiunea din fisier
- *blocare partajata* - cand portiunea partajata poate fi citita simultan de catre mai multe procese, dar nici un proces nu scrie

Blocare concilianta prin **fcntl**

```
struct flock {
...
    short l_type; /*Tip blocare: F_RDLCK,   F_WRLCK, F_UNLCK */

    short l_whence;
    /*Mod interpretare l_start: SEEK_SET, SEEK_CUR, SEEK_END */

    off_t l_start;    /* Offset start regiune */

    off_t l_len;   /* Numar de octeti de blocat/deblocat */

    pid_t l_pid;  /* PID-ul procesului concurent ce a blocat deja                    regiunea  (F_GETLK) */
...
};
```

Blocare concilianta prin **fcntl** (cont.)

#include <fcntl.h>

int fcntl(int fd, int cmd, struct flock* lock);

- cmd: poate lua una dintre valorile:
  - **F_GETLK**
  - **F_SETLK**
  - **F_SETLKW**

man fcntl

Blocare prin **lockf**

#include <unistd.h>

int lockf(int fd, int cmd, off_t len);

- *cmd* - poate lua una dintre valorile:
    - F_ULOCK - deblocarea unei regiuni blocate
    - F_LOCK - blocarea unei regiuni
    - F_TLOCK - testare si blocare daca nu este deja blocata
    - F_TEST - testarea unei regiuni daca e blocata sau nu

man 3 lockf