Sisteme de operare - lab8

Programe C de lucru cu fisiere Unix

Documentație suplimentară

- Standard C I/O
- Tutorial C
- Tutorial C by Brian W. Kernighan. (o prezentare mai detaliată a limbajului C).

Aspecte teoretice

Pentru a putea acționa asupra unui fișier, este nevoie înainte de toate de o metodă de a identifica în mod unic fișierul. În cazul funcțiilor discutate, identificarea fișierului se face printr-un așa-numit *descriptor de fișier* (*file descriptor*). Acesta este un număr întreg care este asociat fișierului în momentul deschiderii acestuia.

Funcțiile open și close

Deschiderea unui fișier este operația prin care fișierul este pregătit pentru a putea fi prelucrat în continuare. Această operație se realizează prin intermediul funcției open:

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

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

Funcția returnează -1 în caz de eroare. În caz contrar, ea returnează descriptorul de fișier asociat fișierului deschis.

Parametri:

- **pathname** - conține numele fișierului
- **oflag** - opțiunile de deschidere a fișierului. Este, în realitate un șir de biți, în care fiecare bit sau grupă de biți are o anumită semnificație. Pentru fiecare astfel de semnificație există definite în fișierul *header* C **fcntl.h** câte o constantă. Constantele se pot combina folosind operatorul '|' (*sau logic pe biți*) din C, pentru a seta mai mulți biți (deci a alege mai multe opțiuni) în parametrul întreg **oflag**.

  Iată câteva din aceste constante:
  - O_RDONLY - deschidere numai pentru citire
  - O_WRONLY - deschidere numai pentru scriere
  - O_RDWR - deschidere pentru citire și scriere
  - O_APPEND - deschidere pentru adaugare la sfârșitul fișierului
  - O_CREAT - crearea fișierului, dacă el nu există deja; dacă e folosită cu această opțiune, funcția **open** trebuie sa primească și parametrul *mode*.
  - O_EXCL - creare "exclusivă" a fișierului: dacă s-a folosit O_CREAT și fișierul există deja, funcția **open** va returna eroare
  - O_TRUNC - dacă fișierul exista, continutul lui este sters
- **mode** - se folosește numai în cazul în care fișierul este creat și specifica drepturile de acces asociate fișierului. Acestea se obțin prin combinarea unor constante folosind operatorul *sau* ('|'), la fel ca și la opțiunea precedenta. Constantele pot fi:
  - S_IRUSR - drept de citire pentru proprietarul fișierului (*user*)
  - S_IWUSR - drept de scriere pentru proprietarul fișierului (*user*)
  - S_IXUSR - drept de executie pentru proprietarul fișierului (*user*)
  - S_IRGRP - drept de citire pentru grupul proprietar al fișierului
  - S_IWGRP - drept de scriere pentru grupul proprietar al fișierului
  - S_IXGRP - drept de executie pentru grupul proprietar al fișierului
  - S_IROTH - drept de citire pentru ceilalti utilizatori
  - S_IWOTH - drept de scriere pentru ceilalti utilizatori
  - S_IXOTH - drept de executie pentru ceilalti utilizatori

Pentru crearea fișierelor poate fi folosita și funcția

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

creat (const char *pathname, mode_t mode)
```

echivalenta cu specificarea opțiunilor O_WRONLY | O_CREAT | O_TRUNC la funcția open.

După utilizarea fișierului, acesta trebuie închis, folosind funcția

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

int close (int filedes)
```

în care filedes este descriptorul de fișier obtinut la open.

Functiile read și write

Citirea datelor dintr-un fișier deschis se face cu funcția

```
#include <unistd.h>

ssize_t read(int fd, void *buff, size_t nbytes)
```

Funcția citește un număr de exact *nbytes* octeți de la poziția curentă în fișierul al cărui descriptor este *fd* și îi pune în zona de memorie indicata de pointerul *buff*.

Este posibil ca în fișier să fie de citit la un moment dat mai puțin de *nbytes* octeți (de exemplu dacă s-a ajuns spre sfârșitul fișierului), astfel că funcția read va pune în buffer doar atâția octeți câți poate citi. În orice caz, *funcția returnează numărul de octeți citiți din fișier*, deci acest lucru poate fi ușor observat.

Dacă s-a ajuns exact la sfârșitul fișierului, funcția returnează zero, iar în caz de eroare, -1.

Scrierea datelor se face cu

```
#include <unistd.h>

ssize_t write(int fd, void *buff, size_t nbytes)
```

Funcția scrie în fișier primii *nbytes* octeți din bufferul indicat de *buff*. Returnează -1 în caz de eroare.

Funcția lseek

Operațiile de scriere și citire în și din fișier se fac la o anumită poziție în fișier, considerată poziția curentă. Fiecare operație de citire, de exemplu, va actualiza indicatorul poziției curente incrementând-o cu numărul de octeți citiți. Indicatorul poziției curente poate fi setat și în mod explicit, cu ajutorul functiei lseek:

```
#include <sys/types.h>
#include <unistd.h>

off_t lseek(int fd, off_t offset, int pos)
```

Funcția poziționeaza indicatorul la deplasamentul *offset* în fișier, astfel:

- dacă parametrul pos ia valoarea SEEK_SET, poziționarea se face relativ la începutul fișierului
- dacă parametrul pos ia valoarea SEEK_CUR, poziționarea se face relativ la poziția curenta
- dacă parametrul pos ia valoarea SEEK_END, poziționarea se face relativ la sfârșitul fișierului

Parametrul *offset* poate lua și valori negative și reprezintă deplasamentul, calculat în octeți.

În caz de eroare, funcția returnează -1.

Functii de biblioteca

Deschiderea și inchiderea unui fișier

```
#include <stdio.h>

FILE *fopen(const char *filename, const char *mode)
```

Funcția deschide fișierul indicat prin *filename*, creează o structură FILE conținând informații despre fișier și returnează un pointer către aceasta. Acest pointer va fi elementul care va putea fi folosit în continuare pentru accesarea fișierului. Parametrul *mode* este un șir de caractere care indica modul de deschidere a fișierului. "r" semnifică deschidere pentru citire, "w" deschidere pentru scriere. Poate fi specificat și tipul fișierului: "t" pentru fișier text, "b" pentru fișier binar. Opțiunile pot fi combinate, de exemplu sub forma "r+t".

Închiderea fișierului se face cu

```
#include <stdio.h>

fclose(FILE *stream)
```

unde *stream* este pointerul spre structura FILE obținut la deschiderea fișierului.

Operatii asupra fișierelor

- **int fprintf(FILE *stream, const char *format, ...);** - scriere în fișier cu formatare; șirul de caractere care specifică formatul este similar celui de la instrucțiunea **printf**.
- **int fscanf(FILE *stream, const char *format, ...);** - citire din fișier, asemanator cu funcția **scanf**.
- **size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);** - citește din fișierul indicat de *stream* un număr de *nmemb* elemente, fiecare de dimensiunea *size*, și le pune în zona de memorie indicata de *ptr*.
- **size_t fwrite(void *ptr, size_t size, size_t nmemb, FILE *stream);** - scrie în fișierul indicat de *stream* un număr de *nmemb* elemente, fiecare de dimensiunea *size*, pe care le ia din zona de memorie indicata de *ptr*.