Operating systems 1, Lecture 6

Sanda-Maria AVRAM, Ph. D.

Leave a feedback

File system calls

- Unix system calls
  - => call directly to the kernel functions of the OS
  - (*the lower level of file processing*)
- Standard functions in C library:
  - **fopen**, **fprintf**, **fscanf**, **fclose**, **fseek**, etc
  - (*the higher level of file processing*)

The *open* system call

```
#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 (with O_CREAT), O_TRUNC

mode:
- access rights; works in accordance with *umask*
- **S_IRUSR, S_IWUSR , S_IXUSR**, *S_IRGRP, S_IWGRP , S_IXGRP*, S_IROTH, S_IWOTH , S_IXOTH

Return: -1 - error
>0 - the file descriptor

man open

The *open* system call. Example 1

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

//example open call
int main(){
        int fisier1 = open("test1", O_CREAT);
        int fisier2 = open("test2", O_TRUNC);
        int fisier3 = open("test3", O_CREAT, S_IRUSR | S_IWUSR | S_IXUSR);
        int fisier4 = open("test4", O_CREAT, S_IRUSR, S_IWUSR, S_IXUSR);
        int fisier5 = open("test5", O_WRONLY);
        return 0;
}
```

Example: Titus Pirtea (ML, 2018)

The *open* system call. Example 2

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

//example open call
int main(){
        int fisier1 = open("cont1", O_WRONLY);
        int fisier2 = open("cont1", O_RDONLY);
        printf("%d and %d", fisier1, fisier2);
        return 0;
}
```

Example: Titus Pirtea (ML, 2018)

The *open* system call. Example 3

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

//example open call with creat command

int main(){
 printf("%s\n", open("fisier1", O_CREAT, S_IRWXU)==-1?"Error":"No error");
 printf("%s\n", open("fisier1", O_CREAT | O_EXCL, S_IRWXU)==-1?"Error":"No error");
 return 0;
}
```

Example: Titus Pirtea (ML, 2018)

The *creat* system call

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

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

/* equivalent to specifying the options: O_WRONLY|O_CREAT|O_TRUNC for the open
function */
```

man creat

The *creat* system call. Example

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

//example creat call

int main(){
     printf("%s\n", creat("fisier1", S_IRWXU)==-1?"Error":"No error");
     return 0;
}
```

Example: Titus Pirtea (ML, 2018)

The *close* system call

```
#include <unistd.h>

int close (int fd) ;

/* fd - file descriptor obtained by open */
```

man close

Functions *read* and *write*

```
#include <unistd.h>

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

Return: 0 – EOF
-1 – error
>0 – number of bytes operated.

man 2 read
man 2 write

Functions *read* and *write*. Example 1

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

// read write example
int main(){
     int fisier = open("cont1", O_RDONLY);
     char buff[256];
     read(fisier, buff, 7);
     write(1, buff, 7);
     return 0;
}
```

Example: Titus Pirtea (ML, 2018)

Functions *read* and *write*. Example 2

```
...
//example of reading a sentence and adding a word
int main(){
     int fisier = open("cont1", O_RDWR);
     char buff[256];
     read(fisier, buff, 1);
     int i = 0;
     while(buff[i++]!='.') read(fisier, buff + i, 1);
     buff[i] = 0;
     write(1, buff, i);
     write(fisier, "Bnb", 3);
     return 0;
}
```

Example: Titus Pirtea (ML, 2018)

Functions *read* and *write*. Example 3

```
...
#include <stdio.h>

// read write example
int main(){
     int fisier = open("cont1", O_WRONLY | O_RDONLY); // not equivalent to O_RDWR
     char buff[256];
     buff[0] = 'A';
     printf("%s\n", read(fisier, buff, 1)==-1?"Error":"No error");
     printf("%s\n", write(fisier, buff, 1)==-1?"Error":"No error");

     return 0;
}
```

Example: Titus Pirtea (ML, 2018)

Function *lseek*

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

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

offset: the number of bytes on which the move is made;

whence: position relative to:
- SEEK_SET - the beginning of the file;
- SEEK_CUR - current position;
- SEEK_END - end of file.

man lseek

Function *lseek*. Example 1

```
// lseek example
int main(){
     int fisier = open("cont1", O_RDWR);

     char buff[1];
     lseek(fisier, 7, SEEK_CUR);
     read(fisier, buff, 1);
     printf("%c", *buff);

     lseek(fisier, -1, SEEK_CUR);
     read(fisier, buff, 1);
     printf("%c", *buff);

     lseek(fisier, 0, SEEK_SET);
     read(fisier, buff, 1);
     printf("%c", *buff);
```

Example: Titus Pirtea (ML, 2018)

Function *lseek*. Example 2

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

int main(){
     int fisier = open("cont2", O_APPEND);
     printf("%ld\n", lseek(fisier, 0, SEEK_SET));
     printf("%ld\n", write(fisier, "A", 1));

     return 0;
}
```

Example: Titus Pirtea (ML, 2018)

Other examples with system calls

```
#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);
# ...
```


Complete example

```
#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;

     /* we open the file */
     fd = open ("gabi.txt", O_RDONLY);
     if (fd < 0) {
          perror("open");
```


Standard functions in the C library

```
#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 ...

System calls *dup* and *dup2*

```
#include <unistd.h>

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

Duplicates the file descriptor **oldfd** as follows:
- **oldfd** and **newfd** refer to the same physical file;
- the access mode is retained from the opening;
- both descriptors share the same current pointer in the file.

Return: -1 - error;
>0 - the new descriptor.

The *dup2* system call. Example 1

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

int main(){
     int fisier1 = open("fisier1", O_WRONLY);
     dup2(fisier1, 1);
     dup2(1, 2);
     printf("Example dup2 \n");
     perror("Example error\n");
     return 0;
}
```

Example: Titus Pirtea (ML, 2018)

The *dup2* system call. Example 2

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

main(){
     int oldfd, newfd;

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

     newfd = dup2(oldfd, 2); /*automatically closes the standard stderr file*/
     /*the new default error file will be ERRORS*/
```


The *fcntl* system call

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

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

Provides or changes properties of an open file;

cmd: F_DUPFD - duplicating a descriptor
F_GETFD (or F_SETFD) – file descriptor flags
F_GETFL (or F_SETFL) – file status flags
F_GETOWN (or F_SETOWN) – manipulation of I/O signals
F_GETLK (or F_SETLK) – blocking files

The *fcntl* system call. Example

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

// fcntl example
int main(){
   int fisier = open("fcntl", O_CREAT | O_WRONLY, S_IRWXU);
   printf("%s\n", (fcntl(fisier, F_GETFL) & O_RDONLY != 0)?"WRONLY":"NOT WRONLY");

   close(fisier);
   return 0;
}
```

Exemple / Teo Petru ML 2018

File Locks

- **advisory lock**
- **mandatory lock**

- *exclusive lock* - when a single process has access to the file or portion of the file;
- *shared lock* - when the shared portion can be read simultaneously by several processes, but no process writes.

Advisory lock by *fcntl*

```c
struct flock {
  ...
  short l_type; /* Lock type: F_RDLCK,   F_WRLCK, F_UNLCK */
  short l_whence; /* Interpretation mode l_start: SEEK_SET, SEEK_CUR, SEEK_END */
  off_t l_start; /* Offset start region */
  off_t l_len; /* Number of bytes blocked/unlocked */
  pid_t l_pid; /* PID of the process that already blocked the region (F_GETLK) */
  ...
};
```


Advisory lock by *fcntl* (cont.)

```c
#include <fcntl.h>

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

cmd: can take one of the values:

- F_GETLK - check whether the indicated resource is locked or not;
- F_SETLK - blocks the resource specified in the third parameter;
- F_SETLKW - blocks only the desired resource. If it's already blocked, it'll wait for its release.

The *lockf* system call

```c
#include <unistd.h>

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

cmd: can take one of the values:

- F_ULOCK - unlock a blocked region;
- F_LOCK - blocking a region;
- F_TLOCK - test and lock if it is not already locked;
- F_TEST - testing a region whether it's locked or not.

The End