2. Operaţii I/O simple

Linux

File descriptor

Întreg ce identifică un fişier în tabela fişierelor deschise de un proces.

File descriptori standard:

- **0** sau **STDIN_FILENO**, standard input
- **1** sau **STDOUT_FILENO**, standard output
- **2** sau **STDERR_FILENO**, standard error

Operaţii pe fişiere

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

`pathname`	calea către fişiere
`flags`	flag-uri de access şi de creare
`mode`	permisiuni la creare
întoarce	file descriptorul creat sau -1 în caz de eroare

```
int close(int fd);
```

`fd`	descriptor fişier
întoarce	0 în caz de succes sau -1 în caz de eroare

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

`fd`	descriptor fişier
`buf`	adresa de început a zonei de memorie pentru citire/scriere
`count`	numărul de octeţi din **buf** solicitaţi pentru citire/scriere
întoarce	numărul de octeţi citiţi/scrişi sau -1 în caz de eroare

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

`fd`	descriptor fişier
`offset`	offset folosit pentru poziţionare
`whence`	poziţia relativă la care se face poziţionarea
întoarce	offset rezultat după poziţionare măsurat relativ la începutul fişierului

```
int unlink(const char *pathname);
```

`pathname`	numele fişierului care va fi şters
întoarce	zer în caz de success, -1 in caz de eroare

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

`oldfd`	vechiul file descriptor
`newfd`	noul file descriptor care va deveni acum o copie a vechiului file descriptor
întoarce	noul file descriptor

Windows

File handle

Identificator pentru un obiect gestionat de kernel-ul Windows.
Device-uri standard:

- **STD_INPUT_HANDLE**, standard input device
- **STD_OUTPUT_HANDLE**, standard output device
- **STD_ERROR_HANDLE**, standard error device

```
BOOL WINAPI SetStdHandle(DWORD nStdHandle, HANDLE hHandle);
```

- **nStdHandle** - device-ul standard pentru care va fi setat handle-ul
- **hHandle** - handle-ul pentru standard device
- *întoarce* - nonzero pentru succes, zero în caz de eroare

```
HANDLE WINAPI GetStdHandle(DWORD nStdHandle);
```

- **nStdHandle** - device-ul standard pentru care va fi obţinut handle-ul
- *întoarce* - handle-ul obţinut sau INVALID_HANDLE_VALUE în caz de eroare

Operaţii pe fişiere

```
HANDLE CreateFile(
    LPCTSTR lpFileName, DWORD dwDesiredAccess,
    DWORD dwShareMode, LPSECURITY_ATTRIBUTES
    lpSecurityAttributes, DWORD dwCreationDisposition,
    DWORD dwFlagsAndAttributes, HANDLE hTemplateFile );
```

- **lpFileName** - numele fişierului creat sau deschis
- **dwDesiredAccess** - GENERIC_READ, GENERIC_WRITE
- **dwShareMode** - FILE_SHARE_READ, FILE_SHARE_WRITE, FILE_SHARE_DELETE
- **lpSecurityAttributes** - de obicei NULL
- **dwCreationDisposition** - CREATE_ALWAYS, CREATE_NEW, OPEN_ALWAYS, OPEN_EXISTING, TRUNCATE_EXISTING
- **dwFlagsAndAttributes** - FILE_ATTRIBUTE_NORMAL, FILE_ATTRIBUTE_READONLY
- **hTemplateFile** - de obicei NULL
- *întoarce* - handle-ul pentru fişier sau INVALID_HANDLE_VALUE în caz de eroare

```
BOOL CloseHandle(HANDLE hObject);
```

- **hObject** - handle care se doreşte a fi închis
- *întoarce* - nonzero pentru succes, zero în caz de eroare

```
BOOL DeleteFile(LPCTSTR lpFileName);
```

- **lpFileName** - numele fişierului care se doreşte a fi şters
- *întoarce* - nonzero pentru succes sau zero în caz de eroare

```
BOOL ReadFile( HANDLE hFile, LPVOID lpBuffer, DWORD
nNumberOfBytesToRead, LPDWORD lpNumberOfBytesRead,
LPOVERLAPPED lpOverlapped );
```

- **hFile** - handle către un fişier deschis
- **lpBuffer** - buffer în care se vor reţine octeţii citiţi din fişier
- **nNumberOfBytesToRead** - numărul de octeţi de citit
- **lpNumberOfBytesRead** - numărul de octeţi efectiv citiţi
- **lpOverlapped** - momentan NULL
- *întoarce* - nonzero pentru succes sau zero în caz de eroare

```
BOOL WriteFile( HANDLE hFile, LPCVOID lpBuffer, DWORD
nNumberOfBytesToWrite, LPDWORD lpNumberOfBytesWritten,
LPOVERLAPPED lpOverlapped );
```

- **hFile** - handle către un fişier deschis
- **lpBuffer** - buffer care se va scrie în fişier
- **nNumberOfBytesToWrite** - numărul de octeţi de scris
- **lpNumberOfBytesWritten** - numărul de octeţi efectiv scrişi
- **lpOverlapped** - momentan NULL
- *întoarce* - nonzero pentru succes sau zero în caz de eroare

```
 DWORD SetFilePointer( HANDLE hFile, LONG lDistanceToMove,
PLONG lpDistanceToMoveHigh, DWORD dwMoveMethod );
```

- **hFile** - handle către un fişier deschis
- **lDistanceToMove** - numărul de octeţi cu care se mută cursorul
- **lpDistanceToMoveHigh** - de obicei NULL
- **dwMoveMethod** - poziţia relativă faţă de care se face mutarea
  - FILE_BEGIN, punctul de start este începutul fişierului
  - FILE_CURRENT, punctul de start este valoarea curentă a cursorului
  - FILE_END, punctul de start este valoarea curentă a sfârşitului de fişier
- *întoarce* - noua valoarea a cursorului în cazul în care **lpDistanceToMoveHigh** este NULL, sau INVALID_HANDLE_VALUE în caz de eroare.