Operating Systems Laboratory 2

fgetc, fgets,fscanf, printf

- **int fgetc(FILE *stream)**

–gets the next character (an unsigned char) from the specified stream and advances the position indicator for the stream.

- **char *fgets(char *restrict s, int n, FILE *restrict stream)**

–reads bytes from stream into the array pointed to by s, until n-1 bytes are read, or a <newline> is read and transferred to s, or an end-of-file condition is encountered. The string is then terminated with a null byte.

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

–reads bytes, interprets them according to a format, and stores the results in its arguments.

- **printf(const char *restrict format, ...);**

–Printf a formatted string on standar output

fread and fwrite

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

–reads nmemb elements of data, each size bytes long, from the stream pointed to by stream, storing them at the location given by ptr.

- **size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream) (man fwrite)**

– writes nmemb elements of data, each size bytes long, to the stream pointed to by stream, obtaining them from the location given by ptr.

- fread() and fwrite() return the number of items successfully read or written (i.e., not the number of characters). If an error occurs, or the end-of-file is reached, the return value is a short item count (or zero).
- fread() does not distinguish between end-of-file and error, and callers must use feof() and ferror() to determine which occurred.

POSIX system calls

- **POSIX system calls**

–open(), read(), write(), close()

–Very low level

–Uses file descriptors instead of FILE* for unbuffered IO

–The STDIO functions are implemented using these functions

- **ssize_t read(int fildes, void *buf, size_t nbyte) (man 2 read)**

–attempt to read nbyte bytes from the file associated with the open file descriptor, fildes, into the buffer pointed to by buf.

–on success, the number of bytes read is returned (zero indicates end of file), and the file position is advanced by this number.

- **ssize_t write(int fildes, const void *buf, size_t nbyte) (man 2 write)**

–attempt to write nbyte bytes from the buffer pointed to by buf to the file associated with the open file descriptor, fildes.

–on success, the number of bytes written is returned (zero indicates nothing was written). On error, -1 is returned

Redirection

>	Redirect standard output
>&	Redirect standard output and standard error
<	Redirect standard input
|	Redirect standard output to another command (pipe)
>>	Append standard output
>>&	Append standard output and standard error

Examples

```
head -n 20 file.txt > file_out.txt
```

copies the first 20 lines of a file to another

```
myApplication >& log.txt
```

makes a log file of myApplication

```
echo ”NEW LINE” >> myFile.txt
```

appends a line to a file

❖Permissions

Permissions

- Permissions types
  - Read(r)
  - Write(w)
  - Exec(c)

- Permissions classes
  - Owner
  - Group
  - Others

Directory permissions

- Read(r)

–File listing

- Write(w)

–Create or delete a file

- Exec(c)

–directory scanning (using cd command)

chmod symbolic mode

`chmod [<who>]<operator><type> filename`

Operators

`+` → add permission

`-` → remove permission

`=` → set permission

Who

`u` → user **`g`** → group **`o`** → other users **`a`** → all

Type → **`r`** , **`w`**, **`x`**

Example

`chmod u+x myfile` → add executable permission for the user to myfile