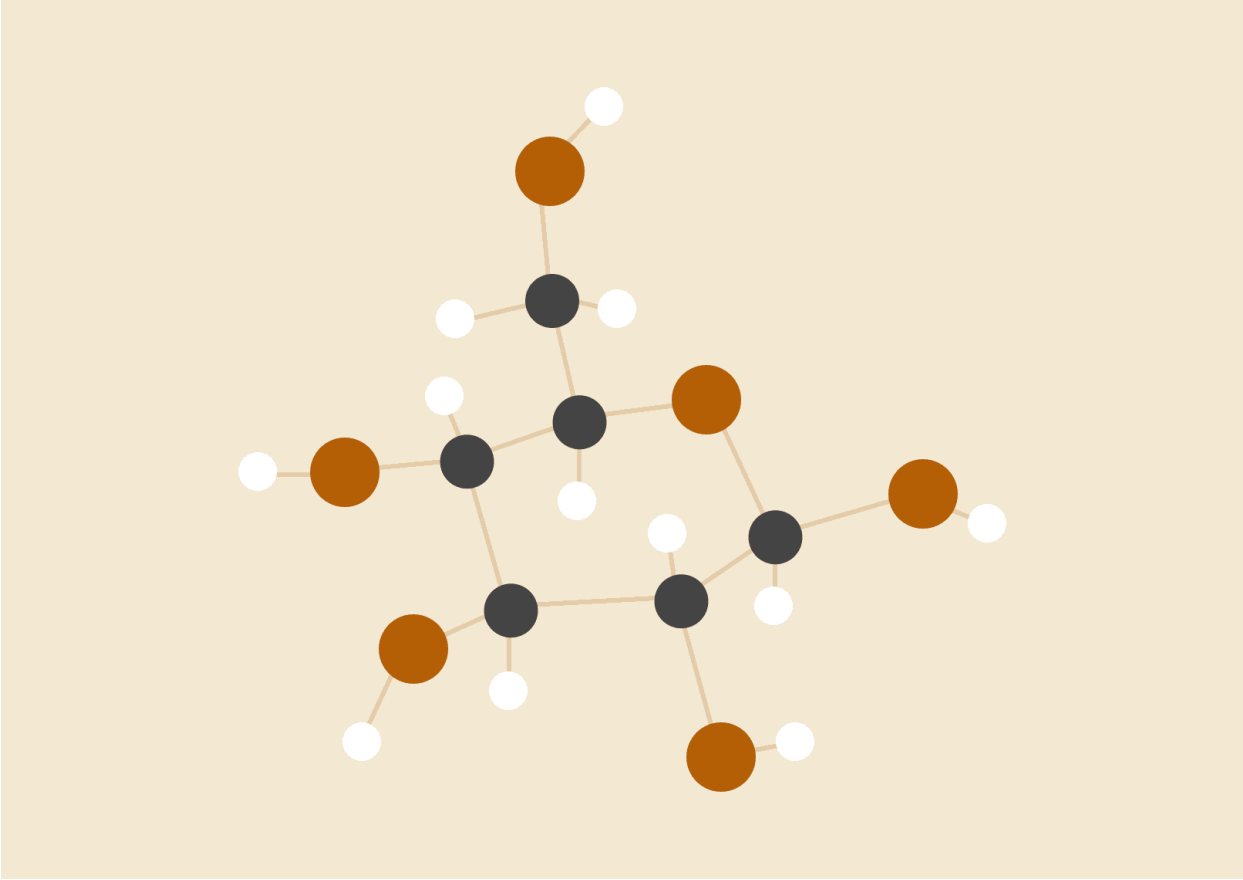
OS LAB REPORT

Course: CS255



Abhiraj Pravin Mengade

24.01.2022

201CS102 IV Sem

FILE MANAGEMENT:

1. creat()

creat : it creates a new file or rewrites an existing one

code:

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

#include <fcntl.h>

int creat(const char *path, mode_t mode);
```

2. open()

open : This call opens or creates a file if not present.

Code:

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

#include <sys/stat.h>

#include <fcntl.h>

int open(const char *path,

                int flags,... /* mode_t mod */);
```

3. close()

close : It closes an open file, returns -1 in case of error.

Code:

```
#include <unistd.h>

      int close(int fd);
```

4. read()

read: This is used to read a certain number of bytes starting from current position.

Code:

```
#include <unistd.h>

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

5. write()

write: This is used to write a certain number of bytes starting from current position.

Code:

```
#include <unistd.h>

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

6. lseek()

Lseek: This call is used to position a pointer in an absolute or relative way.

Code:

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

    #include <unistd.h>

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

7. link()

Link: This call is used to link one file to another directory. I.e it changes the path of the file.

Code:

```
#include <unistd.h>

int link(const char* oldpath, const char* newpath);

int symlink(const char* oldpath, const char* newpath);
```

8. unlink()

Unlink: This call unlinks the link that was created.

Code:

```
#include <unistd.h>

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

9. access()

Access: This call checks whether the calling process can access the file

pathname.

Code:

```
#include <unistd.h>

int access(const char* path, int mod);
```

10. chmod()

Chmod: This system call modifies the access rights of a certain file.

Code:

#include <sys/types.h>

#include <sys/stat.h>

int chmod(const char* path, mode_t mod);

11. chown()

Chown: This call modifies the owner (UID) and the group (GID) of a certain file.

Code:
#include <sys/types.h>

#include <unistd.h>

int chown(const char* path, uid_t owner, gid_t grp);

12. unmask()

Unmask: This call sets the calling process's file mode creation mask

To the value *mask* & 0777 and returns the previous value of the mask.

Code:

#include <sys/types.h>

#include <sys/stat.h>

mode_t umask(mode_t mask);

13. ioctl()

Ioctl: This system call manipulates the underlying device

parameters of special files.

Code:

```
#include <sys/ioctl.h>


    int ioctl(int fd, unsigned long request, ...);
```

PROCESS MANAGEMENT:

1. execl()

execl: This call executes the commands it receives as arguments.

Code:

```
#include <unistd.h>
void  main() {
  char *Path = "/bin/ls";
  char *arg1 = "-lh";
  char *arg2 = "/desktop";
  execl(Path, Path, arg1, arg2, NULL);
}
```

2. fork()

Fork: This call creates another child process running along with the current one.

Code:

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

//creates another child process

fork();

   printf("Hello");

   return 0;

}
```

3. wait()

Wait: This call waits till the child process is complete and then executes the parent one.

Code:

```
#include

#include

pid_t wait(int *stat_loc);
```

4. exit()

Exit: This call terminates the current process.

Code:***void exit (int status);***

5. getuid()

Getuid: Returns the real USER ID (the person who is logged in) of the current process.

Code:

#include <unistd.h>

uid_t getuid(void);

6. geteuid()

Geteuid: Returns the **effective** USER ID of the current process.

Code:

#include <unistd.h>

uid_t geteuid(void);

7. getgid()

getgid: Returns the **real** Group ID of the current process.

Code:

#include <unistd.h>

gid_t getgid(void);

8. getegid()

getegid: Returns the **effective** GROUP ID of the current process.

Code:

#include <unistd.h>

gid_t getegid(void);

9. getpid()

getpid: Returns the **process** ID of the current process.

Code:

#include <unistd.h>

pid_t getpid(void);

10. getppid()

getppid: Returns the **parent process** ID of the current process.

Code:

#include <unistd.h>

pid_t getppid(void);

11. kill()

Kill: This system call can be used to send any signal to any process group or process.

Code:

#include <signal.h>

int kill(pid_t pid, int sig);

12. alarm()

Alarm: This call sets an alarm clock for delivery of a signal

Code:

#include <unistd.h>

unsigned int alarm(unsigned int seconds);

13. chdir()

Chdir: This call changes the working directory of the current process to the specified path.

Code:

```
#include <unistd.h>

int chdir(const char *path);

int fchdir(int fd);
```