Linux Basic Commands

ls- list directory

```
sandhya@ubuntu:~$ ls
Desktop    Downloads      minikube-linux-amd64  Pictures  snap       Videos
Documents  get-docker.sh  Music                 Public    Templates
sandhya@ubuntu:~$
```

---

man –access the manual of command

```
sandhya@ubuntu:~$ man ls
```

```
LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current directory by default).

       Mandatory arguments to long options are mandatory for short options t

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              with -l, scale sizes by SIZE when printing them; e.g., '--bloc

       -B, --ignore-backups
              do not list implied entries ending with ~
```

Press q – to exit

---

cd- Navigate to a specific folder

```
sandhya@ubuntu:~$ cd Documents
sandhya@ubuntu:~/Documents$
```

---

mkdir - Creates a directory

```
sandhya@ubuntu:~/Documents$ mkdir demo_project
sandhya@ubuntu:~/Documents$ ls
C  demo  demo_project  minikube  nw
sandhya@ubuntu:~/Documents$
```

---

touch: Creates a file without opening it for editing.

nano: A simple text editor, suitable for beginners and quick edits.

vim: A powerful and complex text editor, suitable for advanced users.

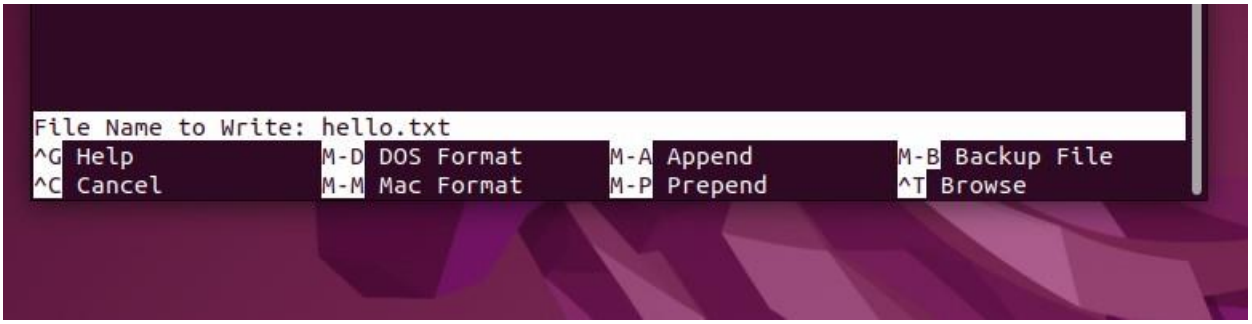
```
sandhya@ubuntu:~/Documents$ cd demo_project
sandhya@ubuntu:~/Documents/demo_project$ nano hello.txt
```

```
  GNU nano 6.2
Hello, this is a demo file.
```

To Save the file-

Press - ctrl x

Press –Y and Enter



---

cat- To display the contents

rm - to remove file
rmdir – to remove directory

System Calls

write- The write system call is used to write data from a buffer to a file descriptor.

```
/demo_project$ man 2 write
/demo_project$
```

```
WRITE(2)                                                                  Li

NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

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

DESCRIPTION
       write() writes up to count bytes from the buffer starting at buf to the file refer

       The number of bytes written may be less than count if, for example, there is insuf
       limit(2)), or the call was interrupted by a signal handler after having written le

       For a seekable file (i.e., one to which lseek(2) may be applied, for example, a re
       actually  written.  If the file was open(2)ed with O_APPEND, the file offset is fi
       performed as an atomic step.

       POSIX requires that a read(2) that can be proved to occur after a write() has retu

       According to POSIX.1, if count is greater than SSIZE_MAX, the result is implementa

RETURN VALUE
       On success, the number of bytes written is returned.  On error, -1 is returned, an
```

To Print text from buffer to Terminal

1. Create and open system_call.c file

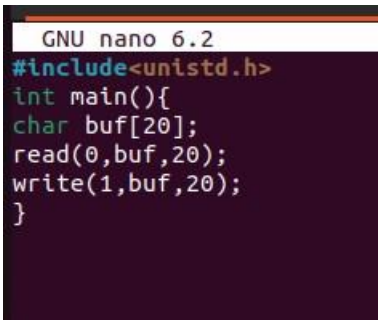
2. After writing the code save it(ctrl-x, y, enter) and execute the file

- **Standard Input (stdin)**: File descriptor 0, used for reading input from the keyboard.
- **Standard Output (stdout)**: File descriptor 1, used for writing output to the screen.
- **Standard Error (stderr)**: File descriptor 2, used for writing error messages to the screen.

---

read - The read system call is used to read data from a file descriptor into a buffer.

1. Write the following code to system_call.c



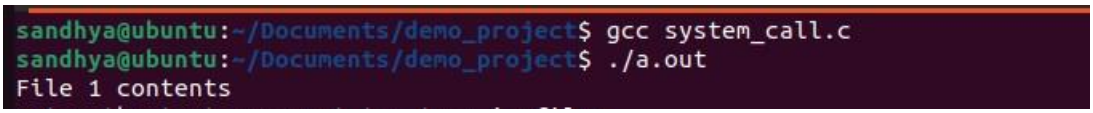
2. Execute

open - The read system call is used to read data from a file descriptor into a buffer.

```
OPEN(2)                    Linux Programmer's Manual                    OPEN(2)

NAME
       open, openat, creat - open and possibly create a file

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

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

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

       int openat(int dirfd, const char *pathname, int flags);
       int openat(int dirfd, const char *pathname, int flags, mode_t mode);

       /* Documented separately, in openat2(2): */
       int openat2(int dirfd, const char *pathname,
                   const struct open_how *how, size_t size);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       openat():
           Since glibc 2.10:
               _POSIX_C_SOURCE >= 200809L
           Before glibc 2.10:
               _ATFILE_SOURCE
```

Task 1: Print the text from file1.txt to terminal

```c
#include <unistd.h>
#include<stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int main(){
char buf[20];
int fd; //declare file discriptor for file 1
fd=open("file1.txt",O_RDWR); //open file 1 in read write mode
read(fd,buf,20); // read from file 1 and store it in buffer
write(0,buf,20); // write to console

}
```



Task 2: Print text from file1.txt to terminal, ask input from the user and store it to file2.txt

SHELL SCRIPTING

echo- to display a text/line.

```
sandhya@ubuntu:~/Documents/bash_project$ echo SANDHYA
SANDHYA
sandhya@ubuntu:~/Documents/bash_project$
```

Create Bash file/Script file: nano hello.sh

```
#!/bin/bash

echo Hello!!!
```

Save it, change the access permission

```
sandhya@ubuntu:~/Documents/bash_project$ nano hello.sh
sandhya@ubuntu:~/Documents/bash_project$ ls -l hello.sh
-rw-rw-r-- 1 sandhya sandhya 27 Jul  5 21:36 hello.sh
sandhya@ubuntu:~/Documents/bash_project$ chmod 744 hello.sh
sandhya@ubuntu:~/Documents/bash_project$ ls -l hello.sh
-rwxr--r-- 1 sandhya sandhya 27 Jul  5 21:36 hello.sh
```

Run it: ./hello.sh

```
sandhya@ubuntu:~/Documents/bash_project$ ./hello.sh
Hello!!!
sandhya@ubuntu:~/Documents/bash_project$
```

To the same above file make changes as follows, save it and run it(./hello.sh)

```
#!/bin/bash

a="hello"
b="world"
c=4
d=5
echo $a + $b
echo $((c+d))
echo Hello!!!
```

Script to run both c program and python file

Note: Create new folder – contains c file, python file, and bash file.

```
#?/bin/bash

src="add.c"
out="outputfile"

gcc -o $out $src

./$out

echo  C Program successfully executed

python3 hello.py

echo  python file successfully executed
```

Change permission:

```
sandhya@ubuntu:~/Documents/bash_project$ nano test.sh
sandhya@ubuntu:~/Documents/bash_project$ ls -l test.sh
-rw-rw-r-- 1 sandhya sandhya 85 Jul  5 21:08 test.sh
sandhya@ubuntu:~/Documents/bash_project$ chmod 744 test.sh
sandhya@ubuntu:~/Documents/bash_project$ ls -l test.sh
-rwxr--r-- 1 sandhya sandhya 85 Jul  5 21:08 test.sh
```

Execute:

```
sandhya@ubuntu:~/Documents/bash_project$ ./test.sh
Calculate SUM
Enter the value of a and b
5 6
Sum=11C Program successfully executed
Hello from python file
python file successfully executed
sandhya@ubuntu:~/Documents/bash_project$
```

Note: Python Installation in Ubuntu

Check version: python3 –version

To install: sudo apt update

sudo apt install python3