

LAB-8.

cut command.

cut from another file.

→ cut -c 2-10 text.txt.

nibsha.

nibbisays

Kaasha.

anada.

oming back

→ date | cut -b 2-5.

onda.

→ date | cut -c 1-6

Monday.

a=\$(date +%H)

if [\$a -lt 23 ^{-a} ~~23~~ ~~\$a~~ -gt 12]

then echo " ~~as~~ evening "

else

echo " morning "

fi

a = \$(date +%H).

if [\$a -gt 11 -a -lt ¹³\$]

then

echo " it is afternoon "

elif [\$a -gt ¹³ -a \$a -lt 20]

then

echo " it is evening "

elif [\$a -gt 20 -a \$a -lt 1]

then

echo " it is night "

else

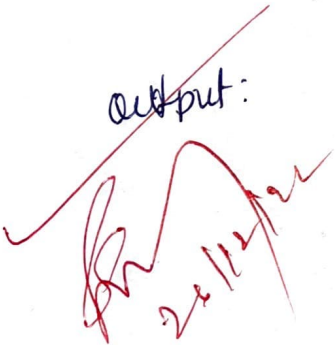
echo " it is morning "

fi

output:

it is afternoon "

Time = 12:16



2/1/23

LAB-9.

Create an employee database with employee details and perform operations.

- ① display header and footer.
- ② print first 5 and last 5 lines
- ③ grep cmd with 5 basic regular expression operations on file
- ④ demo of cut ~~file~~ and column and fieldwise tasks.

① pr -h "head" emp.txt.

2023-01-02 12:28 head page 1

emp ID	name	Salary.
001	anika	1000
002	anik	1500
:	:	:
010	qrs	6000

→ head -n 1 emp.txt.

employee ID Name Salary.

→ tail -n 1 emp.txt

010 qrs 6000

② head -n 5 emp.txt.

Employee ID	name	Salary
001	anishka	1000
002	anik	1500
003	abc	2000
004	bcd	3000

tail -n 5 emp.txt.

006	efg	3500
007	hij	1200
008	klm	5000
009	nop	5700
010	qrs	6000

③ → grep "abc" emp.txt.

003	<u>abc</u>	2000
-----	------------	------

→ grep "[Aa][bB]c" emp.txt.

003	<u>abc</u>	2000
-----	------------	------

→ grep "(Nn)[Aa]" emp.txt.

Employee ID	<u>NAME</u>	SALARY
-------------	-------------	--------

→ grep "[a-a]" emp.txt.

001	anishka	1000
002	anik	1500
003	abc	2000

→ grep "^00" emp.txt.

001	aniksha	1000
002	anik.	1500
003	abc	2000
004	bcd	3000
005	cde	2500
006	efg	3500
007	hi	1200
008	ilm	5000
009	nop.	5400

→ ls -l | grep "^0" emp.txt.

001	aniksha	1000
002	anik.	1500
003	abc	2000
004	bcd.	3000
:		
008	ilm	5000
007	nop	5400
010	qrs	6000

→ grep "[^a-zA]" emp.txt.

001	aniksha	1000
002	anik.	1500
003	abc	2000

→ grep "0.0\$" emp.txt.

004	bcd	3000
008	ilm	5000
010	qrs	6000

② → cat -c 4-6 emp.txt

LO1

→ cat -c 15-30 emp.txt

NAME	SALA
anishha	1000
anik	1500
abc	200
bcd	300
cde	2500
efg	3500
hi	1200
ilm	5000
nop	5700
qrs	6000

→ cat -d " " -b4 emp.txt

2000
3000
2500
3500
1200
5000
5700
6000

-b4

emp.txt

2/1/22

③

④ grep '[0-9]' emp.txt

001	anishha	1000
002	anik	1500
003	abc	2000
007	hi	1200
008	ilm	5000
010	qrs	6000

LAB-10.

① Program with three sort commands.

```
echo " anik / shree / man / pav / ektha " > dum.txt
echo " dum file "
cat dum.txt
echo " "
echo " sorted in reverse "
sort -r dum.txt.
```

```
echo " "
echo " sorted in alphabetical "
sort -b dum.txt.
echo " ----- "
```

```
echo " 40\n23\n45\n67\n71 " > num.txt.
echo " number file "
cat num.txt
echo " "
echo " sort in numerical "
sort -n num.txt.
```

Output:

```
dum file
anik
shree
man
pav
ektha.
```


Sorted in reverse

shree
pav
man
ektha
anik

Sorted in alphabetical

anik
ektha
man
pav
shree.

num file

40
23
45
67
1

sort in numerical

1
23
40
45
67.

② ~~Shell program on paste cmd.~~

echo " anik \n shree \n mans " > dumdum.txt

echo " dumdum file "

cat dumdum.txt

~~echo " "~~

echo " _ _ _ _ "

echo " 40 \n 23 \n 45 \n 67 \n 1 " > num.txt

echo " number file "

cat num.txt.

echo " "

echo ~ after paste ~
paste -d "\t" dumdum.txt num.txt.

Output:

dumdum file

aniksha
shreya
mans

number file

40
23
45
67
1

after past:

aniksha	40
shreya	23
mans	45
	67
	1

③ Demo program for uniq -f.

nano fruit.txt.

apple
orange
grape
grape.

\$ uniq fruit.txt.

apple
orange
grape.

echo " aniksha \n shreeya \n mans \n aniksha " > dum dum.txt

echo " dum dum file "

cat dum dum.txt

echo " _ _ _ _ _ "

echo " changing cases "

tr '[:lower:]' '[:upper:]' < dum dum.txt

Output:

ANIKSHA.S
SHREEYA.
MANS
ANIKSHA.

④ write C/C++ program to that outputs the contents of its environment list.

```
{ #include <stdio.h>  
int main (int argc, char* argv[])  
{  
    int i;  
    char *ptr;  
    extern char **environ;  
    for ( ptr = environ; *ptr != 0; ptr++)  
        printf ("%s\n", *ptr);  
    return 0;  
}
```

→ nano prog.c

{ }

→ gcc prog.c

→ ./a.out

Output = SHELL = /bin/BASH

Session manager = local /bin/mx -HP -Pro -330 -mt !

rest of the environment.

Waw
y-1-20

2-11-2014

APR 2014

MAY

JUN 2014

LAB-11.

① Write a C/C++ program to emulate the Unix ln command.

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

```
int main (int argc, char * argv[])
```

```
{
    if (argc < 3 || argc > 4 || (argc == 4 && strcmp (argv[1], "-s") != 0))
```

```
{ printf ("usage: ./a.out [-s] <orig -file > new -link>\n");
    return -1;
}
```

```
if (argc == 4) {
    if ((symlink (argv[2], argv[3])) == -1)
```

```
printf ("Symbolic cannot create the link\n");
```

```
else
```

```
printf ("symbolic link can be created\n"); }
```

```
else
```

```
{ if ((link (argv[1], argv[2])) == -1)
```

```
printf ("cannot make hard link\n");
```

```
else
```

```
printf ("hard link created\n");
```

```
}
```

```
return 0;
```

```
}
```

Output.

gcc. prog1.c.

→ ./a.out prog1.c.

usage (: ./a.out [-s] <orig-files> <new-link>
(doesn't work).

→ ./a.out prog1.c hard

hard link created.

→ ./a.out -s prog1.c soft.

Symbolic link created.

② Write C/C++ POSIX compliant program that prints
POSIX defined config.

```
# define _POSIX_SOURCE
```

```
# define _POSIX_C_SOURCE
```

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

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

```
int main () {
```

```
#ifdef _POSIX_JOB_CONTROL
```

```
printf ("system does support job control\n");
```

```
#else
```

```
printf ("system does not support job control\n");
```

```
#endif
```

```
#ifdef _POSIX_SAVED_IDS.
```

```
printf ("system supports saved set-UID and saved set-GID\n");
```



```
else  
    printf ("system does not support saved set -uid and saved  
        set -gid\n");
```

```
#endif
```

```
#ifdef _POSIX_CHOWN_RESTRICTED
```

```
    printf ("chown - restricted is %d, _POSIX_CHOWN_REST",
```

```
#else
```

```
    printf ("system does not support chown - restricted option, ");
```

```
#endif
```

```
#ifdef _POSIX_NO_TRUNC.
```

```
    printf ("Pathname %s, _POSIX_NO_TRUNC);
```

```
#else
```

```
    printf ("system does not support system side ");
```

```
#endif
```

```
#ifdef _POSIX_VDISABLE
```

```
    printf ("disable %s, _POSIX_VDISABLE);
```

```
#else
```

```
    printf ("system does not support _POSIX_VDISABLE\n");
```

```
#endif
```

```
return 0;
```

```
}
```

Output

```
gcc prog2.c  
./a.out prog2.c.
```

System supports job control

System supports saved set-uid and saved set-gid

chown - restricted option 0

Pathname truncate option is 1.

Disable character for terminal files is 0.

③ write C/C++ program which demonstrates interprocess communication between a reader process and writer process. Use mkfifo, open, read, write and close API in prog.

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

```
int main(int argc, char ** argv) {
```

```
    int fd;
```

```
    char buf[256];
```

```
    if (argc != 2 && argc != 3)
```

```
    {
```

```
        printf("usage: %s <file> [<arg>] \n", argv[0]);
```

```
        return 0;
```

```
    }
```

```
    mkfifo(argv[1], S_IRFIFO | S_IRWXU | S_IRWXG | S_IRWXO);
```

```
    if (argc == 2) {
```

```
        fd = open(argv[1], O_RDONLY | O_NONBLOCK);
```

```
        while (read(fd, buf, sizeof(buf)) > 0)
```

```
            printf("%s", buf);
```

```
    }
```

```
    else
```

```
    {
```

```
        fd = open(argv[1], O_WRONLY);
```

```
        write(fd, argv[2], strlen(argv[2]));
```

```
    }
```

```
    close(fd);
```


Output:

gcc prog3.c

./a.out

aaditya

filename.

"im saying hello"

message.

→ in a new terminal

→ ./a.out aaditya.

im saying hello

Done
11/1/24