TECHNO INDIA
SILIGURI INSTITUTE OF TECHNOLOGY
A Satyam Roychowdhury Initiative
NAAC ACCREDITED

UNIT 6: Customization and Filters

BCAN 601: UNIX and Shell Programming

Table of Content

Environment Variable

- When you run a command, the shell makes out in shell variables and their values available to the program.
- The program can use this information to customize its actions.
- The collection of variables and values provided to programs is called the environment.
- Your environment includes variables set by the system, such as HOME LOGNAME, and PATH.
- You can display your environment variables with the command env.

Environment Variable

- HOME: Contains the absolute path name of your login directory. HOME is automatically defined and set to your login directory as part of the login process. The shell itself uses this information to determine the directory to change when you type cd with no argument.
- LOGNAME: Contains your login name. It is automatically set by the system.
- PWD: is a spatial variable that gets set automatically to your present working dietary. You can use this variable to include your current directory in the prompt or in a command line.

Environment Variable

- PATH: Path lists the directories in which the shell searches to find the program to run when you type a command. A default path is set by the system, but many users modify it to add additional command directories.
- The typical example of customized PATH, for user usr is
- $PATH = \$PATH{:}/sbin{:}\ /usr/bin{:}\ usr/sbin$
- It is also common to create a subdirectory called bin in your home directory. Instead of adding every directory that contains a command or executable to your path, you can create symbolic link to the command in your directory.

Environment Variable

- PS1 defines your prompt. The default value is $.
- PS2 defines your secondary prompt and has a default value of >.
- Most users like to customize the prompt by adding information such as the current working directory.
- $\$PS1 = \$LOGNAME\ \$PWD >$

Environment Variable

- TERM is used by vi and other screen-oriented programs to get information about the type of terminal you are using. This information is necessary to allow the program to match their output to your terminal's capabilities, and to interpret.

Command Aliases

- Aliases are very convenient features introduced in csh and supported by tcsh, ksh and bash.
- A command aliases is a word linked to a block of text that is substituted by the shell whenever that word is used as a command.
- You can use aliases to give command names that are easier for you to remember or to type and to automatically include particular options when you run a command.
- The syntax of defining aliases varies slightly according to the shell you are using.
- $\$\, alias \lg ls\ - g$

Command History

- Most modern shell keep a list of all the commands you enter during a session.
- The history list can be used to review the command you have recently entered or to repeat commands you have used.
- You can display a list of previously entered comments with the history command
- The following is a typical history list displays
- $\$ \, history$

$grep$ families

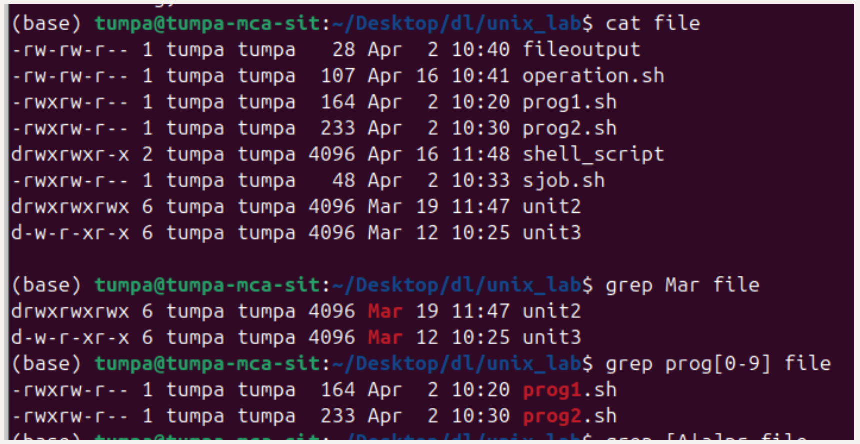
- The most commonly used unique tools for finding words in files are $grep, fgrep, egrep$. These commands searches a target or pattern that you specify.
- You can use them to extract information from files to search output of a command for lines relating to a particular item and to locate files containing a particular keyword
- The three commands in grape family are very similar. All of them print lines matching up target.

$grep$ families

- $grep$: is the most commonly used of the three commands. It lets you search for a , which may be one or more words or patterns containing wildcards and other regular expression elements.
- $fgrep$: Does not allow regular expression but does allow to search for multiple targets.
- $egrep$: Takes a richer set of regular expressions as well as allowing multiple target searches and is considerably faster than $grep$.

$grep$ command

- $grep$ command searches through one or more files for lines containing a target and then point all of the matching lines it finds.
- The following command print all lines in the file mtg_note that contains the word “room”.
- $\$\ grep\ room\ mtg_{note}$
- $room$: target string will be searched
- mtg_note: name of the file where to search
- If the target contains spaces, you have to enclose it in quotes to prevent the shell from treating different wars as separate arguments.



```
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cat file
-rw-rw-r-- 1 tumpa tumpa   28 Apr  2 10:40 fileoutput
-rw-rw-r-- 1 tumpa tumpa  107 Apr 16 10:41 operation.sh
-rwxrw-r-- 1 tumpa tumpa  164 Apr  2 10:20 prog1.sh
-rwxrw-r-- 1 tumpa tumpa  233 Apr  2 10:30 prog2.sh
drwxrwxr-x 2 tumpa tumpa 4096 Apr 16 11:48 shell_script
-rwxrw-r-- 1 tumpa tumpa   48 Apr  2 10:33 sjob.sh
drwxrwxrwx 6 tumpa tumpa 4096 Mar 19 11:47 unit2
d-w-r-xr-x 6 tumpa tumpa 4096 Mar 12 10:25 unit3

(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ grep Mar file
drwxrwxrwx 6 tumpa tumpa 4096 Mar 19 11:47 unit2
d-w-r-xr-x 6 tumpa tumpa 4096 Mar 12 10:25 unit3
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ grep prog[0-9] file
-rwxrw-r-- 1 tumpa tumpa  164 Apr  2 10:20 prog1.sh
-rwxrw-r-- 1 tumpa tumpa  233 Apr  2 10:30 prog2.sh
```

Pattern using regular expression

- Grape allows you search for patterns that may match a number of different words or string

Symbol	Definition	Example	Matches
.	Matches any single characters	$th.nk$	$think, thank$
\	Quotes the following character	$script\backslash.py$	$script.py$
$*$	Matches zero or more reputation of the previous item	$ap ** le$	$aple, apple$
[]	Matches any one of the characters inside	$[Ee]arth$	$Earth, earth$
$[a-z]$	Anyone of character ranges	$[0-9]*$	$0, 1, 9, 19$
^	Matches the beginning of a line	$\hat{}if$	Any line beginning with if
$	Matches the end of a line	\.$	Any line ending with dot(.)

Options for grep

Options	Descriptions
$-i$ (ignore case)	Find all lines containing a target regardless of upper case or lower case distinctions
$-r$ (recursive search)	Recursively search files in all the subdirectories of the current directory
$-n$ (line number)	Allows you to list line number on which the target is found
$-l$ (supress match lines)	Supress the printing of matching lines and just print the names of the file that contain the target
$-v$	Print all lines that do not contain the specified target

$fgrep$ command

- $fgrep$: it is similar to grep, but with three main differences:
    - ➢ It search for several targets at once
    - ➢ It does not allows use of regular expression to search for a pattern
    - ➢ Faster than grep
- With $fgrep$ you can search for lines containing any one of several targets.
- When you give $fgrep$ with multiple search targets each one must be on a separate line and the entire search stream must be in quotation mark.

$ $fgrep$ "$anik$
$radha$
ram" phone_list

$egrep$ command

- $egrep$ comment is most powerful number of the grape common family
- $egrep$ search for several targets in two ways: by putting them on separate lines as $fgrep$ or by separating them with vertical bar or pipes.
- $egrep$ "anik|radha|ram" $phone\_list$

egrep command

Symbol	Definition	Example
+	Matches one or more repetition of the previous term	.+
?	Matches previous item zero or one times	Ind\(.html)?
()	Group a portion of the pattern	Script(\.py)?
|	Matches either the value before or after the |	(E|e)xit

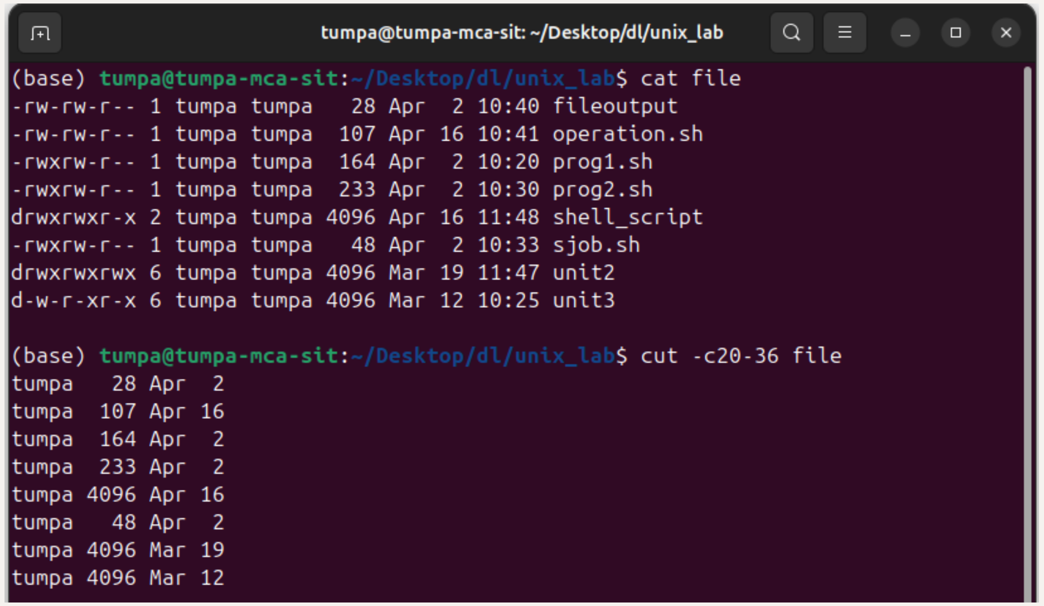
Working with columns and fields

- cut allows you to select particular columns or fields from files
- $colrm$ deletes one or more columns from a file or set of files
- Paste glues together columns or fields from existing files
- Join merges information from two database files.

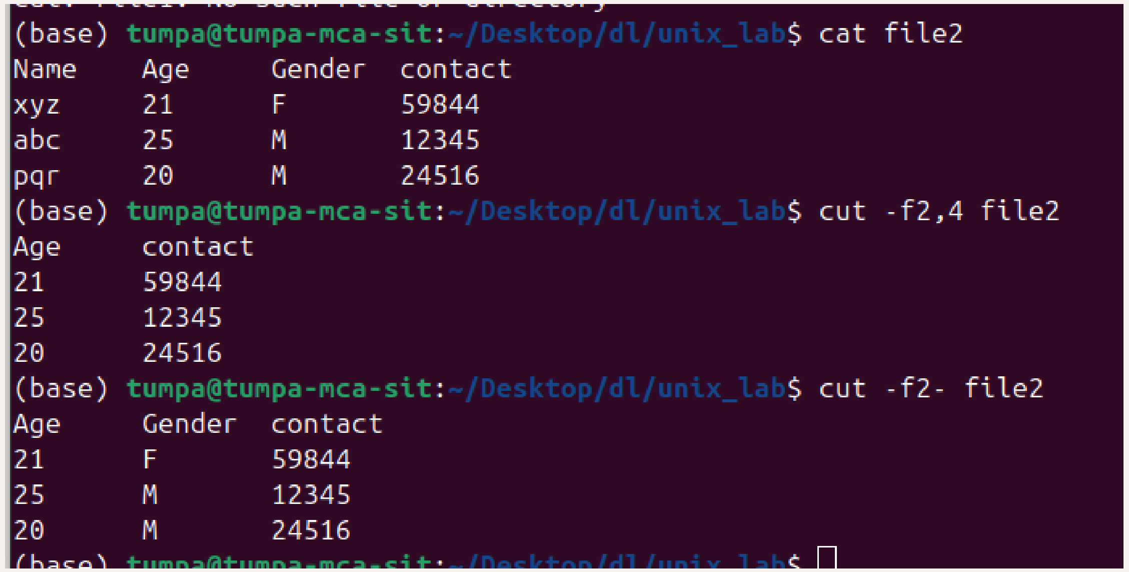
cut command

- When you use $cut,$ you have to tell it how to identify the field and which field to select.
- You can identify fields either by character position or by the use of field separator characters.
- You must specify either the $-c$ or $-f$ option and the field to select.

cut command



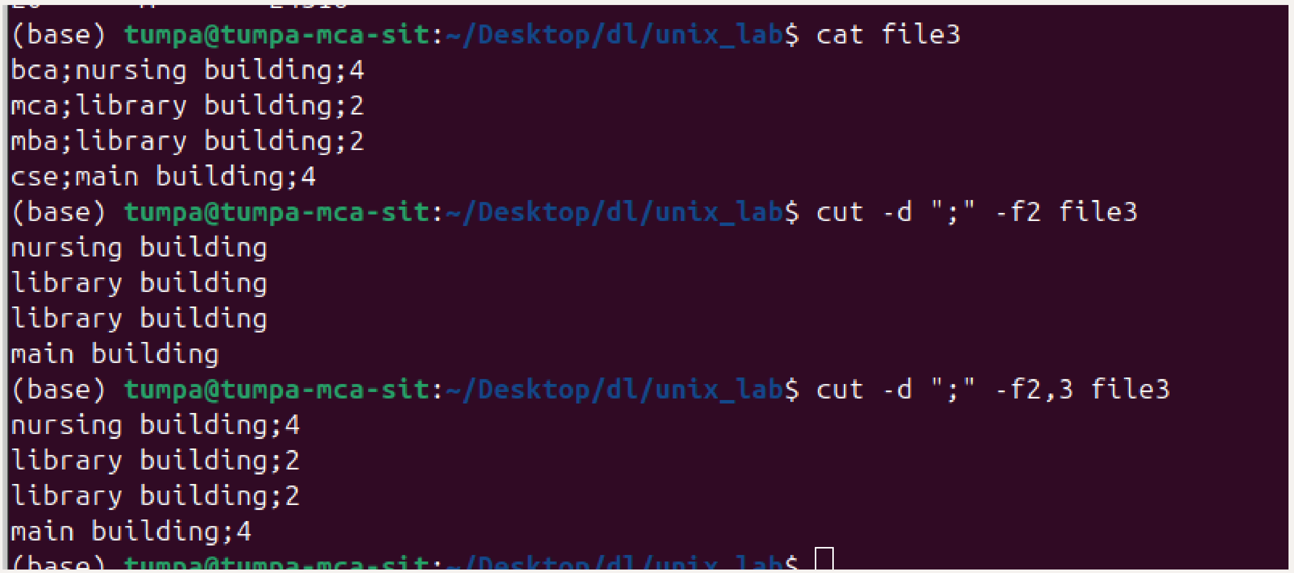
```
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cat file
-rw-rw-r-- 1 tumpa tumpa   28 Apr  2 10:40 fileoutput
-rw-rw-r-- 1 tumpa tumpa  107 Apr 16 10:41 operation.sh
-rwxrw-r-- 1 tumpa tumpa  164 Apr  2 10:20 prog1.sh
-rwxrw-r-- 1 tumpa tumpa  233 Apr  2 10:30 prog2.sh
drwxrwxr-x 2 tumpa tumpa 4096 Apr 16 11:48 shell_script
-rwxrw-r-- 1 tumpa tumpa   48 Apr  2 10:33 sjob.sh
drwxrwxrwx 6 tumpa tumpa 4096 Mar 19 11:47 unit2
d-w-r-xr-x 6 tumpa tumpa 4096 Mar 12 10:25 unit3

(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cut -c20-36 file
tumpa   28 Apr  2
tumpa  107 Apr 16
tumpa  164 Apr  2
tumpa  233 Apr  2
tumpa 4096 Apr 16
tumpa   48 Apr  2
tumpa 4096 Mar 19
tumpa 4096 Mar 12
```



```
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cat file2
Name     Age      Gender   contact
xyz      21       F        59844
abc      25       M        12345
pqr      20       M        24516
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cut -f2,4 file2
Age      contact
21       59844
25       12345
20       24516
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cut -f2- file2
Age      Gender   contact
21       F        59844
25       M        12345
20       M        24516
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$
```

Specifying delimiters

- fields are separated by delimiters. Sometimes fields are separated by some other separators like comma, semicolon etc.
- To tell cut to treat some other character as the field separator, use the $-d$ delimiter option followed by the character.



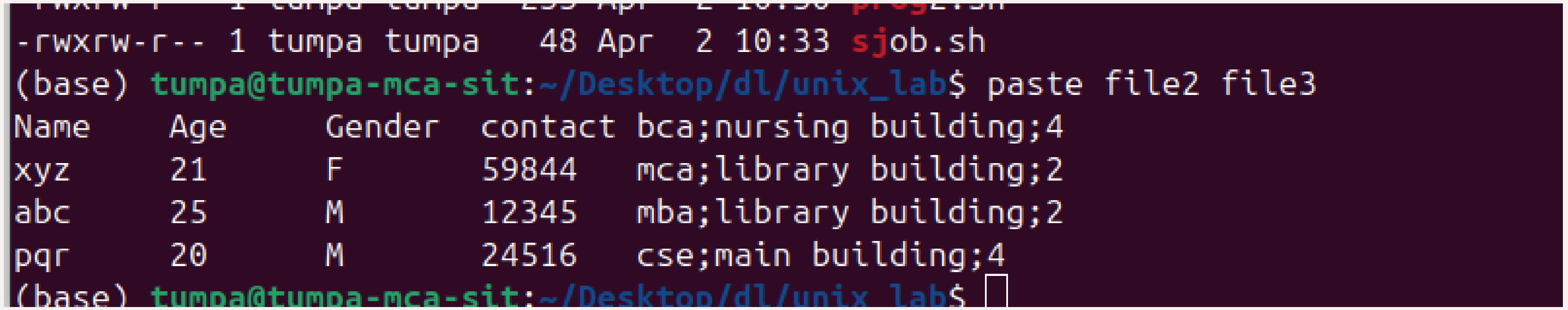
```
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cat file3
bca;nursing building;4
mca;library building;2
mba;library building;2
cse;main building;4
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cut -d ";" -f2 file3
nursing building
library building
library building
main building
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cut -d ";" -f2,3 file3
nursing building;4
library building;2
library building;2
main building;4
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$
```

colrm command

- The *colrm* Command is a specialized command that you can use to remove one or more columns from a file or set of files.
- $cat\ note|colrm\ 8\ 12$
- The command deletes the characters in columns 8-12 from the file note.

paste command

- paste command joins files together line by line.
- You can use it to create new tables by gluing together fields or column from two or more files.



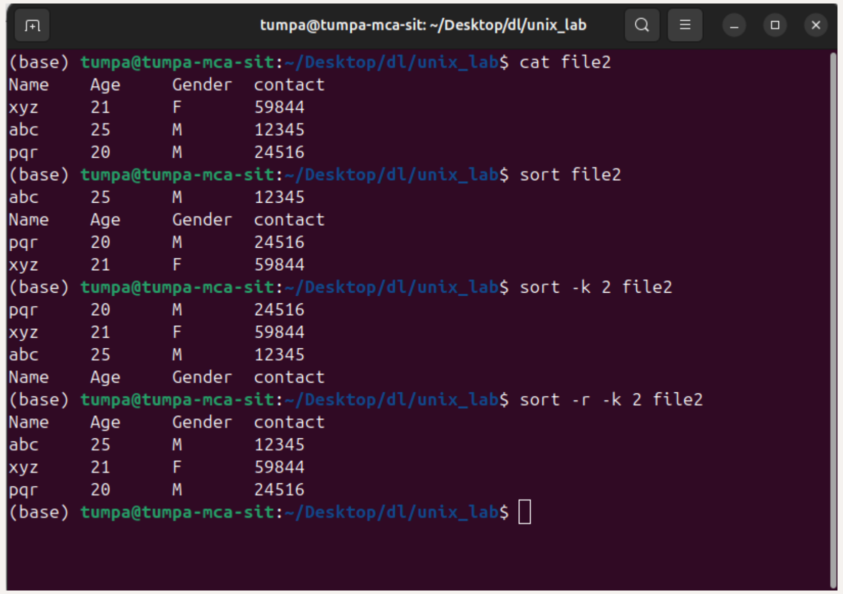
```
-rwxrw-r-- 1 tumpa tumpa   48 Apr  2 10:33 sjob.sh
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ paste file2 file3
Name    Age     Gender  contact bca;nursing building;4
xyz     21      F       59844   mca;library building;2
abc     25      M       12345   mba;library building;2
pqr     20      M       24516   cse;main building;4
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ 
```

pr command

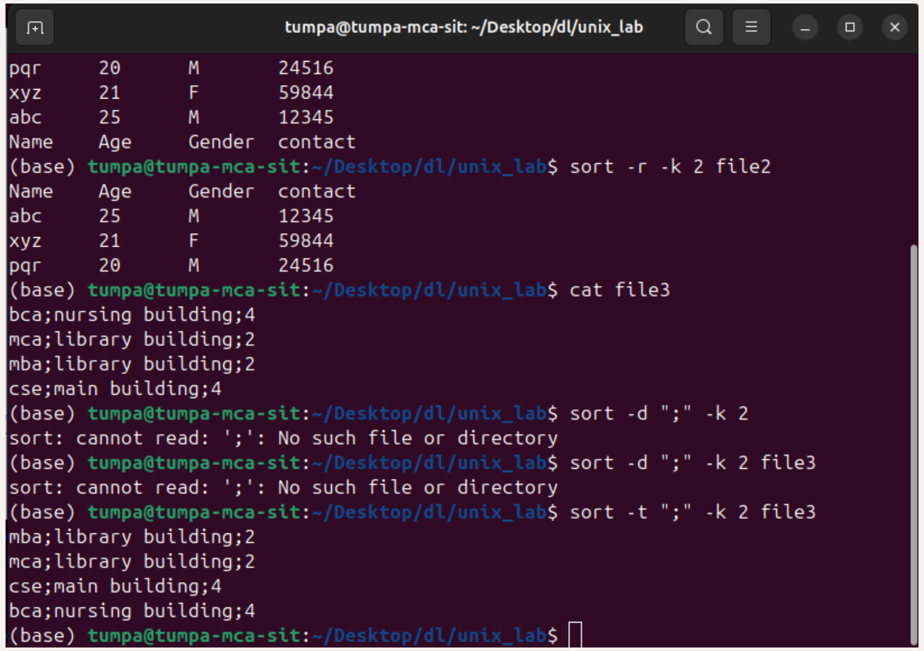
- The most common use of pr command to add header to every page of a file.
- pr command actually adds 5 lines of margin both at the top and bottom of the page. The header part shows the date and time of the last modification of the file with the file name and the page number.
- $pr\ [option]\ filename$

Sort command

- Sorting is the ordering of data in ascending or descending order.
- *sort* command orders a file.



```
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cat file2
Name    Age     Gender  contact
xyz     21      F       59844
abc     25      M       12345
pqr     20      M       24516
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ sort file2
abc     25      M       12345
Name    Age     Gender  contact
pqr     20      M       24516
xyz     21      F       59844
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ sort -k 2 file2
pqr     20      M       24516
xyz     21      F       59844
abc     25      M       12345
Name    Age     Gender  contact
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ sort -r -k 2 file2
Name    Age     Gender  contact
abc     25      M       12345
xyz     21      F       59844
pqr     20      M       24516
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$
```



```
pqr     20      M       24516
xyz     21      F       59844
abc     25      M       12345
Name    Age     Gender  contact
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ sort -r -k 2 file2
Name    Age     Gender  contact
abc     25      M       12345
xyz     21      F       59844
pqr     20      M       24516
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ cat file3
bca;nursing building;4
mca;library building;2
mba;library building;2
cse;main building;4
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ sort -d ";" -k 2
sort: cannot read: ';': No such file or directory
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ sort -d ";" -k 2 file3
sort: cannot read: ';': No such file or directory
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$ sort -t ";" -k 2 file3
mba;library building;2
mca;library building;2
cse;main building;4
bca;nursing building;4
(base) tumpa@tumpa-mca-sit:~/Desktop/dl/unix_lab$
```

$uniq$ command

- $uniq$ command filters or removes repeated lines from files. It is usually used with files that first have been sorted by sort.
- $\$\ sort\ names.*\ |uniq > name$

tr command

- tr replaces a set of characters to another set of characters.
- $\$\ tr\ [option]\ set1\ set2$
- $\$\ cat\ file1|tr\ [a-z]\ [A-Z]$

```
ID,Name,Department,Salary
101,Alice,HR,50000
102,Bob,IT,60000
103,Charlie,Finance,55000
104,David,IT,62000
105,Eva,Marketing,48000
106,Frank,HR,51000
107,Grace,Finance,53000
108,Hannah,Marketing,47000
109,Ian,IT,61000
110,Jane,HR,52000
```

Try yorself

- Find all employees in the "IT" department.
- Find all entries where the name starts with the letter "A".
- Display only the names of the employees.
- Extract and display Name and Salary columns only.
- Sort the file based on Salary in ascending order.
- Sort names alphabetically.
- Find all employees in "HR"
- Extract only their names
- Sort the names alphabetically