

Text Processing

Starts at 9:05 p.m.

Agenda

1. Arrays
2. grep
3. sed
4. awk
5. regex
6. Program

Arrays

① Declaring an array.

```
my_array = ( "apple" "banana" "orange" )  
my_array = ( 1 2 3 4 5 )
```

② Accessing an array element

```
echo ${my_array[0]}  
      → apple  
echo ${my_array[1]}
```

→ banana

echo \${my_array[@]}

↳ access all elements of an array.

using *

→ apple banana orange

→ \$# size of array.

"\${numbers[@]}"

↓

arrays/ "1" "2" "3" "4" "5"

$\$@$ → treats each element as argument
"1" "2" "3"

$\$*$ → treats entire array as argument.

"1 2 3 4 5"

→ grep.

→ grep -flag "pattern" filename.

-i -> case insensitive

apple Apple -i

-n -> prints line number along with the text

-c -> counts the matches

-w -> match whole word

-B -> before

-A -> after

-C -> around

-v -> Invert match

-o -> only the matched part, instead of the whole line.

-E → extended regular expression.

grep -E 'lemon|tart' text

SED Stream editor.

search

replace

print

delete

insert

append.

① substitution (s)

② deletion (d)

③ Printing (p)

① substitution. (s)

sed 's/original/replace/' file

↓

replaces first occurrence on each line.

sed 's/original/replace/g' file

↙

globally

-i → in place update.

(modify the contents of file)

sed '2 s/apple/biryani/' file

sed 's/apple/biryani/I,' file

↓

case Insensitive.

apple → biryani
APPLE → biryani

g

② Deleting.

sed ' /pattern / d ' file

sed ' 2d ' file → for deleting a line

③ Inserting & Appending

sed ' 3i \ Inserted line ' file

sed ' 3a \ Appended line ' file

④ sed for printing. (p)

sed -n ' 2p ' file

↳

suppressing all other lines

sed -n '2,4p' file

sed -n '/apple/p' file

↓

for pattern matching.

Break → 10:30 pm.

AWK.

↗ fields

	C ₁	C ₂	C ₃	C ₄
Records. ↙	R ₁			
	R ₂	Q		
	R ₃			
	R ₄			

\$ → fields.

\$2 →

NR == 2

default separator → space



-F ' , '

awk ' NR == 2; {print \$4}'



-F ' , '

→ pattern matching. → print \$0

awk '/pattern/ {print \$}' text

→ changing field separator.

awk -F ' ' '{print \$2}' awk_delim

→ Range Patterns.

fetch logs from pattern 1
to pattern 2

★ awk ' /start/ , /END/' filename.

start

{ }

end

→ Conditional Patterns

awk ' \$4 == "apple" { print } ' file

awk '\$2=\$2*2 {print}' numbers

→ Built in functions.

awk '{ print tolower(\$0) }' file

Regular Expressions.

Regular Expressions.

grep app * expression
app \$

awk

sed

① • dot → any character in place of dot.
- w

grep ^ "a.p" text

matches → aip
a2p
abp
:

② ^ & \$

^ → position at start of line

\$ → position at end of line

grep " ^apple" text

grep "LE\$" text

③ Character class []

grep [Aa]pple text

[a-z] → denotes all words

[A-Z] → " " capital

[0-9] → all numbers

[^] → inside brackets is used for negation.

④ Quantifiers

* → zero or more occurrences

+ → 1 or more occurrences.

grep "ap*" file

ap p*
└─┘
needed

p* → 0 will also be fine.

grep "ap+" file

└─┘
needed.

$\{n\} \rightarrow$ match exact number of times
 $\{n, \}$ \rightarrow atleast n times
 $\{n, m\} \rightarrow$ Between n & m times

string literals .

\backslash \rightarrow Escape character.

grep "\." file

Q Fetch emails from a file using regular expressions.

____ @ ____ . ____

a-z \rightarrow all lowercase

A-Z \rightarrow all uppercase.

0-9 \rightarrow all numbers.

.

%

+

$\wedge [a-zA-Z0-9._ \% + -] + @$

$[a-zA-Z0-9._] + . [a-zA-Z]{2,} \$$

end

.com ✓
.in ✓

grep -E "^[a-zA-Z0-9._ \% + -] + @[a-zA-Z0-9._] + \.[a-zA-Z]{2,} \$" email

↳ .com → .ac .in
→ .in

→ example @ gov. ac . in ✓

- `^[a-zA-Z0-9._ \% + -] +`:

- `[]`: This denotes a character class.

- `[a-zA-Z]`: Matches any uppercase or lowercase letter.

- `[0-9]`: Matches any digit.

- `.`: Matches a literal dot.

- `_`: Matches an underscore.

- `%`, `+`, `-`: Matches these special characters.

- `+`: This quantifier means "one or more" of the preceding characters. So, this part matches the local part of the

email (like `user.name`, `user-name`, `user123`, etc.).

1. `@`: This matches the `@` symbol, which separates the local part from the domain.

2. `**[a-zA-Z0-9.-]+\`**`:

- Similar to the local part, this part matches the domain name, allowing letters, digits, dots, and hyphens.

- `**+\`**`: Again, means "one or more" characters.

3. `**\.`**`: Matches a literal dot (.). Since `\.`` is a metacharacter in regex (matching any character), we escape it with a backslash to treat it as a normal character.

4. `**[a-zA-Z]{2,}\`**`:

- This matches the top-level domain (like `.com``, `.net``, `.org``, etc.).

- `**{2,}\`**`: Means "at least 2 characters" (to allow for domains like `.com`` and `.org``).

5. `**$**`: Asserts the end of the string, ensuring that nothing follows the email address.

→ History from system.
SED

sed '2d' text

32 sed '2d' text

33 sed '2p' text

34 sed -n '2p' text

35 sed 's/apple/banana/' text

37 sed 's/apple/banana/' text

38 sed 's/apple/banana/g' text

39 sed -i 's/cherry/guava/g' text

42 sed 's/apple/banana/g' text

43 sed 's/apple/banana/gl' text

44 sed 's/\bapple\b/banana/gl' text

45 sed '/apple/d' text

46 sed '3i\Inserted line' text

47 sed '3a\Inserted line' text

48 sed -n '/apple/p'text

49 sed -n '/apple/p' text

171 sed 's/apple/guava/' text

173 sed 's/apple/guava/' text

174 export apple=\$(sed 's/apple/guava/' text)

218 sed 's/cake/biryani/' text

219 sed 's/apple/biryani/' text

220 sed 's/apple/biryani/g' text

222 sed -i 's/water/biryani/g' text

224 sed -i 's/biryani/tea/g' text

226 sed '2 s/apple/biryani/g' text

227 sed '/oranges/d' text

228 sed '2d' text

229 sed '3i\Inserted line' text

230 sed '3a\Inserted line' text

231 sed '2p' text

232 sed -n '2p' text

233 sed -n '2,4p' text

235 sed -n '2,4p' text

236 sed -n '/apple/p' text

249 sed -n "/apple/p" text

318 history | grep sed

Awk.

51 awk '\$2 > 50 {print}' numbers

52 awk '\$2 > 50 {print \$0}' numbers

53 awk '{print \$0}' numbers

54 awk '{print}' numbers

58 awk '{print \$3}' numbers

59 awk '{NR==2;print \$3}' numbers

60 awk 'NR==2{print \$3}' numbers

62 vi awk_delim

63 awk -F ',' 'NR==2{print \$3}' awk_delim

64 awk -F ',' '{print \$3}' awk_delim

65 cat awk_delim

71 awk '/apple/,/oranges/' text

72 awk '/apple/,/mangoes/' text

73 awk '/guava/,/mangoes/' text

77 awk '\$4=="Apple"{print}' text

78 awk 'BEGIN {IGNORECASE=1} /apple/ {print}' text

79 awk '\$2 > 50 {print \$0}' text

80 awk '\$2 > 50 {print \$0}' numbers

81 awk '{ \$2 = \$2 * 2; print \$0 }' numbers

240 cat awk_delim

245 cat awk_delim

247 awk '/apple/ {print}' text

251 awk '{print \$2}' numbers

252 cat awk_delim

253 awk -F ' ' '{print \$2}' awk_delim

255 awk -F 'guava' '{print \$2}' text

256 awk -F 'guava' 'NR==1;{print \$2}' text

257 awk '{print \$2}' awk_delim

258 cat awk_delim

259 awk '{print \$1}' awk_delim

260 awk '\$4=="apple"{print}' text

264 awk '\$4=="apple"{print}' text

266 awk '\$2>50{print}' numbers

267 awk '\$2=\$2*2 {print}' numbers

314 `grep -E "ERROR" data | awk '{print $1 $2 $3}'`

315 `grep -E "ERROR" data | awk '{print $1 " " $2 " " $3}'`

316 `grep -E "ERROR" data | awk '{print $1 " " $2 " " $3 " "for (i=5; i<=NF; i++) printf $i }'`

317 `grep -E "ERROR" data | awk '{print $1 " " $2 " " $3 " " for (i=5; i<=NF; i++) printf $i }'`

319 `history | grep awk`