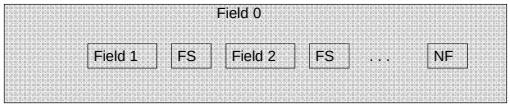
EECS2031

AWK

AWK

FILE

RECORD



AWK -- INTRODUCTION

- **AWK consists of "awk" , instructions (in quotes or a file), and an input file**
- **Instructions consist of pattern and action**
- **Pattern, a statement or expression , regular expressions enclosed in forward slashes**
- **Actions: one or more statement separated by semicolon or new lines**
- **awk 'pattern' filename**
- **awk '{action}' filename**
- **awk 'pattern {action}' filename**

HOW SED WORKS

- Examples
- awk '/mary/'
- awk '{print $1}'
- awk '/Mary/{print $1, $2}'
- awk '/Mary/{print $1 $2}' file (what is the difference?
- date
- Mon Mar 28 11:08:14 EDT 2016
- date | awk '{print "Month: " $2 "\nYear: " $6}'
- Month: Mar
- Year: 2016

SED COMMANDS

- AWK contains two special patterns: BEGIN and END. Both are given without slashes.
- The BEGIN pattern specifies actions to be performed before any records are processed:
- BEGIN {action}
- The END pattern specifies actions to be performed after all records are processed:
- END {action}
- awk –F: ' ' file the input field separator is ":"

NR AND NF

- NR record number
- NF number of fields in a record

awk '{print "Record " NR "has " NF "fields and ends with " $NF}' file

Not it's time for
All good men to
Come to the help of
Their party

Record 1 has 4 fields and ends with for
Record 2 has 4 fields and ends with to
Record 3 has 5 fields and ends with of
Record 6 has 2 fields and ends with party

BEGIN -- END

```
# Program to print a file with a header and footer.
BEGIN       { print "Beginning of file";
              print "-----------------" ;
              }
// # Print every line in the file.
END { print "------------"; print "End of file."}
```

COMPARISON

- <, <=, >, >=, ==, !=
- ~ and !~ match and doesn't match

- awk '$3 ~ /Bill/{print $3}' file
- awk '#8 ~ /[0-9][0-9]$/ {prin $7}' file
- awk '$3 > 5000{print $1}' file
- Conditional expressions
- awk '{max=($1 > $2) ? $1 : $2; print max}' file
- awk '$3 * $4 > 500' file
- Awk '$2 == "CA"{print $1, $2}' file

LOGICAL OPERATORS

- && || !
- awk '$2 > $5 && 2 <= 15' file
- Range
- awk '/Bill/,/Suzanne/' fie

CONDITIONAL OPERATOR

- awk '{print ($7 > 4) ? "high "$7 : "low "$7
- awk '$3== 'Chris"{$3="Christian"; print}' file

VARIABLES

- name = "John" string
- number = 35 number
- to change from string to number name +0
- To change from number to string number " "
- All fields and array elements created by the split function are considered strings

- BEGIN {digits = "^[0-9]+$" }
- $2 ~ digits
- Will print all lines where second field is a string of digits
- index("banana", "an") returns 2
- match(s,r) finds the leftmost longest substring in the string s that is matched by the regular expression r and returns the index where the substring begins, or 0 if no match
- split(s,a,fs) splits the string s into array a according to the field separator fs
- gsub(r,s) substitute s for r globally in $0 (gsub(r,s,t)