The **awk** Command

**awk** is a simple file-processing language

- **awk [-Fc] [-f progsfile] [prog] [var=val] [file ...]**
  - `-Fc` set c as the input field separator
  - `-f progsfile` read program from progsfile
  - `prog` command line program: '{ prog }'
  - `var=val` set awk variable to value
  - `file` input file
  - `BEGIN` and `END` mark special programs segments executed only once (at the beginning or ending of the program)
  - Programs can have line matching conditions
  - Programs can have line addresses
  - Built in variables
    - NR, NF — number of records and number of fields
    - FS, IFS, OFS — field separator (input and output)
    - $0, $1, $2, etc. — individual fields; $0 is the entire input line
File Processing

**awk** details

- **awk** processes line-oriented text
  - piped in
  - from file(s) specified on the command line

- The program is executed for each line of text (an implied loop)

- An entry from the passwd file
  - csnort:x:314:314:Cranston Snort:/home/csnort:/bin/csh
  - the field separator is :
  - $0 is: csnort:x:314:314:Cranston Snort:/home/csnort:/bin/csh
  - $1 is: csnort
  - $5 is: Cranston Snort

- **awk** can do simple math, including floating point
**awk Examples**

Simple, one-line programs

- `awk '{ print $2, $1}' file`
  - print fields 1 & 2 of each line in reverse order separated by a space
  - if input looks like: firstName lastName
  - output will be: lastName firstName

- `awk 'length > 72 { print $0 }' file`
  - print lines longer than 72 characters; $0 is the entire line

- `awk -F : '$1 == "csnort" { print $1, $5 }' /etc/passwd`
  - set field separator to : and print fields 1 and 5 if field 1 is exactly “csnort”; read input from the file /etc/passwd
  - compare with the example on the following page
**awk and Shell Scripts**

A confusing reuse of variable names

- $1, $2, ... $9
  - denote positional parameters in a shell script
  - denote fields in an `awk` program

- Interpret as a field if between the single quotation marks

- The following example
  - “id” is an `awk` variable, which is initialized to the first parameter passed to the shell script
  - prints the name associated with the given id

```bash
#!/bin/bash

awk -F: '{if ($1 == id) print $1, $5}' id=$1 /etc/passwd
```
More `awk` Examples

BEGIN / END example

# print fields in reverse order
# NF is the awk notation for number of fields

`awk { for (i = NF; i > 0; --i) print $i } file`

# find the average of the numbers in field (column) 1
# BEGIN section is executed once, before any data is read
# END section is executed once, after all data is read
# middle sections is executed for each line of data read

`awk 'BEGIN { sum = 0 }
   { sum += $1 }
   END { avg = sum / NR; print avg }' file`
Even More awk Examples

Simple databases with awk

File .checkbook contains
   deposit::8/15/97:200.00
   418:Albertsons:8/16/97:43.27

awk -F: 'BEGIN { total = 0 }
    { if ($1 == "deposit")
        total += $4
    else
        total -= $4 }
END { print total }' $HOME/.checkbook

File .phonebook contains
   Albert Einstein:123 My Street:555-1212

awk -F: '{ if ($1 == name)
        print $1, $2, $3 }' name="$1" $HOME/.phonebook