Solaris Utilities

nawk Introductory Example

- *nawk* is the most complex individual command we are going to introduce you to on this course.
    - * There are complete books on it!
- "You can do almost anything with *nawk*"
    - * Yes you can, but there may be other better ways
- What does *nawk* actually do?
    - * Reads input line by line.
    - * Checks each line to see if it meets specified conditions, and if it does, performs specified actions, for example:

$ ***nawk '/^[^#]/{print "Host "$2" Address "$1}'*** \
/***etc/hosts***

```
Host didcot     Address 192.168.200.2
Host carlisle   Address 192.168.200.3
Host ash        Address 192.168.200.4
Host hunt       Address 192.168.200.5
Host gatwick    Address 192.168.200.6
Host cod        Address 192.168.200.7
Host perch      Address 192.168.200.8
Host stanstead  Address 192.168.200.9
Host golfer-gw  Address 192.168.200.10
etc....
```

- Takes all lines in the file */etc/hosts* which are not comments, and displays the workstation name and the Internet (IP) address.

Solaris Utilities

nawk Introductory Example (contd.)

- Let's look at this in more detail.
- The SECOND parameter (easiest one first!) is the input file; in this example, we have chosen */etc/hosts* which is the file that defines the names and addresses of all workstations known on the network.
    - * If no input file is given, then *nawk* reads from *stdin* - ideal for piping information in from another tool.
- The FIRST parameter contains the *nawk* statements.
- These statements, as in any programming language, tell *nawk* what actions to perform, and are known as the *nawk* program.
    - * *nawk* programs can be quite complex ...

      /^[^#]/{print "Host "$2" Address "$1}
    - * *nawk* programs are almost always going to need single quote shell protection (because they almost always contain $ characters) and will probably also contain many other special characters as well - even our first example contains space, [,], {, }, ", and $.

Solaris Utilities

nawk Introductory Example (contd.)

- The *nawk* program consists of one or more statements of the form:

  pattern { action }

- In our example, the pattern was ***/^[^#]/*** - a regular expression!
- (The regular expression characters are delimited by the “/” at each end.)
- Specifically, it calls for the following action to be taken on all input lines which do not start with a # character - i.e. which are not comment lines in file */etc/hosts*.

  * OTHER PATTERNS LATER

- Our sample action was ***print "Host "$2" Address"$1***

  This statement instructs *nawk* to print out

  * The constant text *Host*

  * The second field from the line just read

  * The constant text *Address*

  * The first field from the line just read

  * Finally a new line character will be output

- As you may guess, this output is routed to '*stdout*'

Solaris Utilities

Simple nawk example

- *nawk* has a number of sensible defaults, so you can write very effective commands much shorter than our first example
- If no action is given, lines that match are sent to stdout unchanged.

  * Example:

```
$ ls  -l  utilities | nawk  '$5>2000'
-rw-r--r--   1 sa2     other     33864         Jan 17  1995 datafile_full
-rw-r--r--   1 sa2     other     21346         Jan 17  1995 datafile_part
-rw-r--r--   1 sa2     other     56394         Nov 22  1993 ex_data1
-rw-r--r--   1 sa2     other     56234         Nov 22  1993 ex_data2
-rw-r--r--   1 sa2     other     57034         Nov 22  1993 ex_data3
-rw-r--r--   1 sa2     other      2051         Nov 22  1993 ex_data4
etc.

$
```

  * Uses *nawk* to print details of those files in the *utilities* directory which are larger than 2000 bytes in size.
  * All lines output by the *ls* command are processed, as no pattern matching was applied.

Solaris Utilities

nawk - printing selected fields

- As we saw in the initial example, it is easy to define the fields to be printed.
- Here is a further example, taking input from an *ls -l* command:-

$ ***ls -l /etc | nawk '{print "file "$9" size "$5}'***

```
file TIMEZONE       size 12
file acct           size 512
file aliases        size 14
file asppp.cf       size 360
file auto_home      size 92
file auto_master.orig     size 83
file autopush       size 16
file chroot         size 18
file clri           size 16
file crash          size 16
file cron           size 16
file cron.d         size 512
file datemsk        size 472
file dcopy          size 17
file default        size 512
etc
$
```

* A TAB character was printed before the word "size". One file name (*auto_master.orig*) was very long, so a tab stop was over-run.

* The *printf* statement (later!) can be used instead of *print* to achieve better formatted output.

Solaris Utilities

Exercise

- Use *nawk* to print the filename, owner and size from an *ls -l* listing.
- Use *nawk* to print out a table from the file *datafile_part*, (under the *utilities* directory) showing:

  * User name

  * Workstation name

  * Date logged in

  * Length of login

Solaris Utilities

nawk - Arithmetic Operations

- *nawk* has a calculation capability
  - * Arithmetic calculations can be written straight into a *print* action ...

```
$ ls  -l | nawk  '{print $9,"       size ",$5/1024,"Kb"}'
datafile      size  1.06934 Kb
fyle          size  0.511719 Kb
here          size  0.12793 Kb
one           size  0.5 Kb
stdcode       size  123.292 Kb
there         size  0.12793 Kb
three         size  0.5 Kb
today         size  0.0283203 Kb
two           size  0.5 Kb
$
```

- Arithmetic operators include

+	add
-	subtract
*	multiply
/	divide
%	remainder when divided by
( )	for changing order of precedence

- Other function such as *sqrt, sin, cos* and *log* are also available within *nawk*, as is a random number generator *rand.*
- A list of functions appears in the *nawk* manual page.

Solaris Utilities

nawk - Variables

- If you want to SAVE the result of a calculation for printing out later, you may do so using a *nawk* variable.
- A ’variable’ is a named memory location in the computer - you choose the name, the computer works out where to store it:

```
$ ls -l | nawk  \
    '{print $9,"      ",runtot=runtot+$5," accumulated size"}'
datafile            1095        accumulated size
fyle                1619        accumulated size
here                1750        accumulated size
one                 2262        accumulated size
stdcode             128513      accumulated size
there               128644      accumulated size
three               129156      accumulated size
today               129185      accumulated size
two                 129697      accumulated size
$
```

- In this example, we chose the descriptive name *runtot* for the variable; within the *print* statement, we have said:-
    - * "*runtot* becomes the old value of *runtot*, PLUS the fifth field on the input line"
    - * Because this calculation is within the *print* statement, the value is also printed out.
- *nawk* assumes that any new variable starts at zero.