UNIX SYSTEMS PROGRAMMING

By

Dr. Trilok Nath Pandey

Dept. of C.S.E

S ‘O’ A, Deemed to be University

ITER, Bhubaneswar.

Previous Class

IO Redirection

- Filters
- – wc
- – sort
- – head
- – tail
- – grep
- – pipe
- – tee

Today's Agenda

- Introduction to Unix Shell and Pattern Matching
- Shell Script
- Interactive Shell Scripts
- Using Positional Parameters
- Using read
- Performing Arithmetic Operations using expr

What is Shell?

- A Shell
- Is a program that interprets commands
- It allows a user to execute commands
- Either manually at a terminal or automatically in programs called shell scripts
- A shell is not an operating system
- It is a way to interface with the operating system and execute commands
- Login shell is BASH = Bourne Again Shell

Bash

- Bash is a shell written as a free replacement to the standard Bourne Shell (/bin/sh)
- Bourne Shell originally written by Steve Bourne for UNIX systems
- It has all the features of the original Bourne Shell, plus additions that make it easier to program
- Bash is a Free Software
- It has been adopted as the default shell on most Linux systems

Advantage of using Shell

- 1. File name short hands
- – Can pick up a whole set of file names as arguments to a program by specifying a pattern for the names
- – Shell will find all the filenames that match the pattern
- 2. Input – Output redirection
- – Can take input from a file and can redirect output to a file
- 3. Personalizing the environment
- – Can define own commands and short hands

Bash

- Shell provides a mechanism for generating a list of file names that match a pattern
- – Example: **ls -l *.c**
- – All file names in the current directory that end in *.c*
- – The character * is a pattern that will match any string including the null string
- – This mechanism is useful both to save typing and to select names according to some pattern

Wild cards

- Wild cards – Special characters interpreted by shell

Command	What does the command do?
*	Matches any number of characters including none
?	Matches a single character
[ijk]	Matches a single character either i , j or k
[!ijk]	Not i, j or k
[x-z]	At least a single character within this ASCII range
[!x-z]	Not a single character within this range
[a-dx-z]	At least a single character within a to d or x to z ASCII range

Wild card – “*”

- * stands for zero or more characters.
- **Examples**:
- – **ls a** output is only a if a is a file, output is the content of a if a is a directory
- – **ls ar*** all those that start with ar
- – **ls *vict** all those files that end with vict
- – **ls *[ijk]** all files having i ,j or k as the last character of the filename
- – **ls *[b-r]** all files which have at least any character between b and r as the last character of the filename
- – **ls *** all files

Wild card – “?”

- ? matches a single character
- **Examples**:
- – **ls a?t** matches all those files of three characters with a as the first and t as the third character and any character in between
- – **ls ?oo** matches all three character files whose filename end with oo

Examples

Command	What does it do?
ls ??i*	Matches any number of characters but definitely two characters before i followed by any number of characters
ls ?	Matches all the single character files
ls *?	Matches any file with at least 1 character
ls ?*	Matches any file with at least 1 character
ls “*”	Matches any file with * as the filename (exactly 1 character which is *)
ls “*”*	Matches all files starting with * as filename

Meta characters

- Characters that have a special meaning to the shell are called Meta characters
- – Examples: < > * ? | &
- Any character preceded by a \ is quoted and loses its special meaning, if any
- – Example:
- echo \? will echo a single ?
- echo \\ will echo a single \
- – \ is convenient for quoting single characters.
- – For quoting more than one character \ is clumsy and error prone
- A string of characters may be quoted by enclosing the string between single quotes

Meta characters

- Example
- – echo xx'****'xx will display xx****xx
- The quoted string may not contain a single quote but may contain new lines, which are preserved
- This quoting mechanism is the most simple and is recommended for casual use
- A third quoting mechanism using double quotes is also available that prevents interpretation of some but not all meta characters

Shell Meta characters

Meta characters	Purpose
>	prog > file; Direct standard output to file
>>	prog >> file; Append standard output to file
<	prog < file; Take standard input from file
|	p1 | p2; Connect standard output of p1 to standard input of p2
<<str	Here document: standard input follows, upto next str on a line by itself
*	Match any string of 0 or more characters in file names
?	Match any single character in file names
[ccc]	Match any single character from ccc in file names
[a-z]	Match any single character from a to z or 0 to 9
;	Command terminator. p1;p2 does p1 then p2
&	Like ; but does not wait for p1 to finish

14

Shell Meta characters

Meta characters	Purpose
`...`	Run command(s) in ...; output replaces `...`
(...)	Run command(s) in ... in a sub shell
{...}	Run command(s) in ... in current shell [rarely used]
$1, $2, etc	$0 to $9 replaced by arguments to shell file
$var	Value of shell variable var
${var}	Value of var, avoids confusion when concatenated with text
\	\c take character c literally, \newline discarded
'...'	Take ... literally
"..."	Take ... literally after $, `...` and \ interpreted
#	If #starts word, rest of line is a comment
var=value	Assign to variable var
p1 && p2	Run p1; if successful, run p2
p1 || p2	Run p1; if unsuccessful, run p2

15

THANK YOU