Unix Tools

Ian Pratt

Michaelmas Term 1999

4 Lectures

no exam questions

Unix Tools

- Widely available
  - even for Windows
  - “standardised”
- Often quicker/more powerful than GUI approaches
  - but, takes time to learn
- Good for ‘medium complexity’ operations
  - in between interactive tasks and real programming
- Course aims:
  - provides a basic introduction
  - let you know where to get further info

Outline

- Lecture 1 : The shell
  - tool philosophy
  - streams, redirections and pipes
  - use of regular expressions
- Lecture 2 : Development tools
  - Make
  - RCS
  - job control
- Lecture 3 : Data Manipulation Tools
  - sed, tr, diff, etc.
- Lecture 4 : Scripting languages
  - intro to perl

Unix Tools Philosophy

Suite of “simple” tools that can be combined to perform complex functions

- Individual tools:
  - address a limited class of problem
  - but, completely general within that class
  - small, “easy to learn”, well implemented
  - easily linked with other tools
- Typical syntax:
  - command <options> <file(s)>
  - take input from stdin or file(s)
  - options start with a `-' or `--‘
  - return an exit code

Getting help:

- man *command*
- *command* --*help* or -*?*
- info pages
- Books:
  - “Unix in a Nutshell”, Daniel Gilly, O-Reilly
- reading other shell scripts:
  -

The Unix Shell

- Ordinary user-mode application
  - user can choose/re-implement shell
- Many to choose from:
  - sh, bsh, csh, ksh, ash, msh, bash, ...
- Two main family lines
  - csh and bsh
- csh & its derivatives are NOT recommended
- GNU bash is ubiquitous and pretty good
- Interactive and shell script use
  - [illegible]
  - [illegible]

Streams, redirects and pipes:1

- – c
- Equivalent to:
  - – c
- Can also typically do:
  - – c
- Use > to append to output file
  - – e
- Use te to get a copy
  - – g

Streams, redirects and pipes:2

- Standard error stream
  - Programs typically send error output to stderr
  - [illegible]
  - Stderr normally doesn't pass through pipe
- Merging stderr to stdout
  - [illegible]
  - [illegible]
- `Here documents'
  - Embedding input in scripts

Command line expansion

- Shell performs expansion before execution
  - Expand wildcards to match filenames
    - `ls` is equivalent to `ls`
    - List all but `hidden' file, including directories
    - `ls`
  - ? character matches any single character
    - `ls`
  - Backslash and quotes can be used to `escape' special characters

Environment variables

- variable_name = value; export value
  -
  -
  -
- Expanded by shell before execution of cmd
  - or syntax
  -
- Use as a `shorthand’
  -
- Enables easy changing later
- lists all env variables defined

Arguments to shell scripts

- $1 … $9 are set to arg1, arg2…arg9
- $0 is the name used to invoke the script
- $# is the number of the last argument
- $* is a list of all arguments (excluding $0)
  - $1 $2 $3 $4 $5 $6 $7 $8 $9

Functions

- Can be defined interactively or in a script
  - [illegible]
  - [illegible]
- Used as shorthand
- Can make scripts more readable

```
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
}
```

Regular expressions : grep

-
- Common options:
  - -i case insensitive
  - -v invert sense of matching
  - -c just produce a count of matching line
  - -l print name of file in which match occured
- egrep necessary for more complex regexps

Regexps

- (A) Parenthesis of a sub-string
- A | B Alternation
- A* Accepts zero or more instances of A
- A+ Accepts one or more instances of A
- A? Accepts zero or one instances of A
- A{n,m} Accepts from n to m repetitions
- [a-z] Match single char e.g. [^0-9] [a-z,A-Z]
- ^ and $ Match start/end of line
- \< and \> Match start or end of words

- `escape' characters of significance to shell!

Finding files : fd

- Find <pathnames> <options>
- options narrow the set of file that match:
  - n (like shell regexps)
  - f: Files, directories
  - n
  - + : at most, exactly, at least
  - e: NOT, OR, AND (AND usually implicit)
  - n Print the results (also -e

Command exit codes

- Zero for success
- Non-zero implies some abnormal condition
- Accessed via $?
- ||and && enable conditional execution
  - Shortcircuting evaluation
- A || B B executed iff A has non-zero exit
- A && B B executed iff A has zero exit code

Expression evaluation: test

- Comparing strings and integers (in bash)
  - [illegible]
  - [illegible]
  - [illegible]
- Testing files for existence, access permissions, modification times, etc.
  - [illegible]File exists and is readable/writeable/executable
- Abbreviated to [illegible]
  - [illegible]

Conditional execution

- [illegible]

- [illegible]

fandestatements

- [illegible]
- [illegible]

Shell Job Control

- make in background at low pri
- bring to the foreground
- suspend it…
- …and put back into background

- list the current background jobs in this shell
- kill job number 3
- list all processes on machine (aux Unix specific…)
- show real-time machine status
- used to start tasks at future times

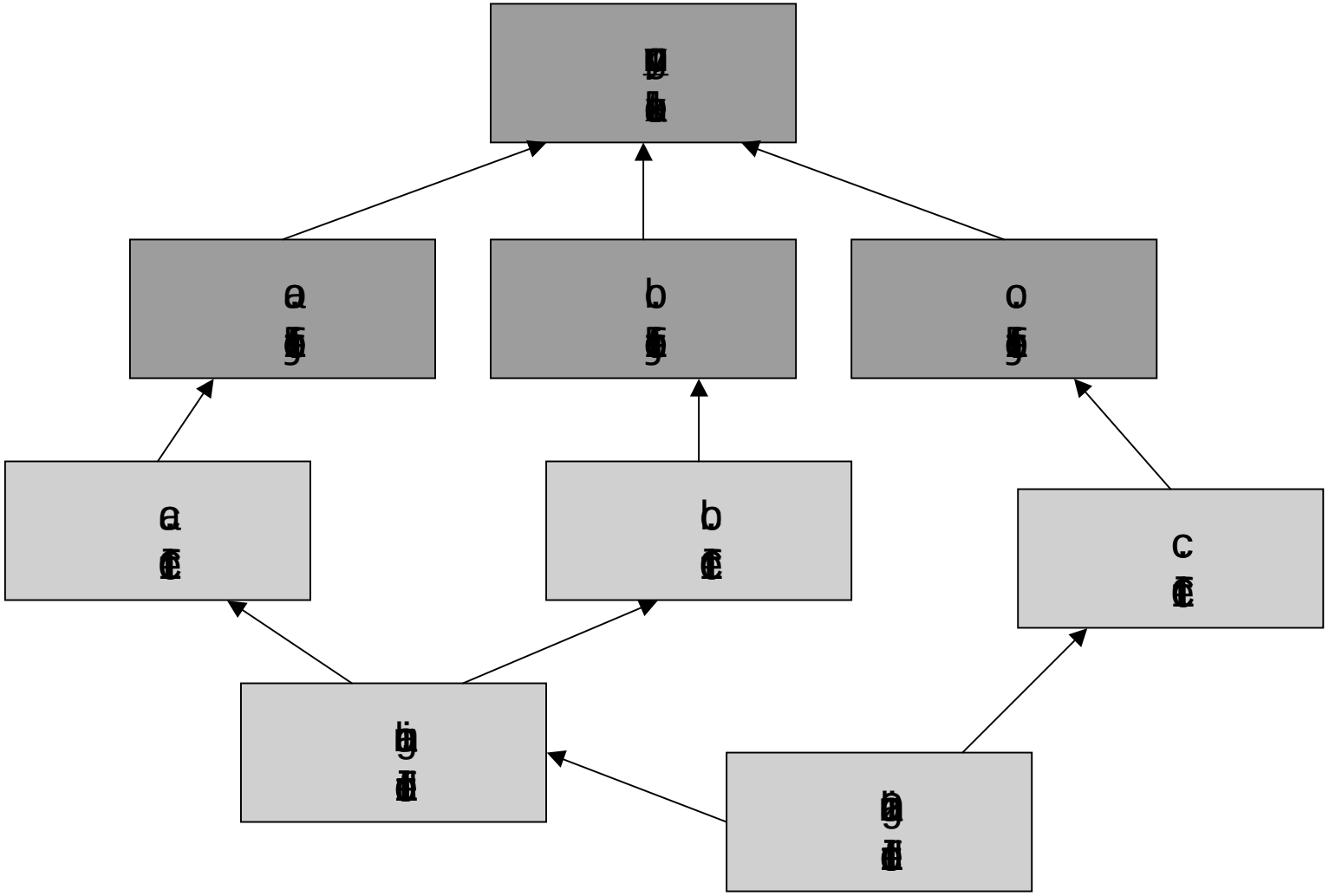
Hand project building

- Automates the build process
  - ‘Documents’ build procedure
  - Can also use to invoke / regression test program, operate revision control system, install, tidy etc.
  - Useful for projects other than program sources
    - E.g. LaTeX documents
- Re-build only what needs updating
  - Exploit parallelism where available

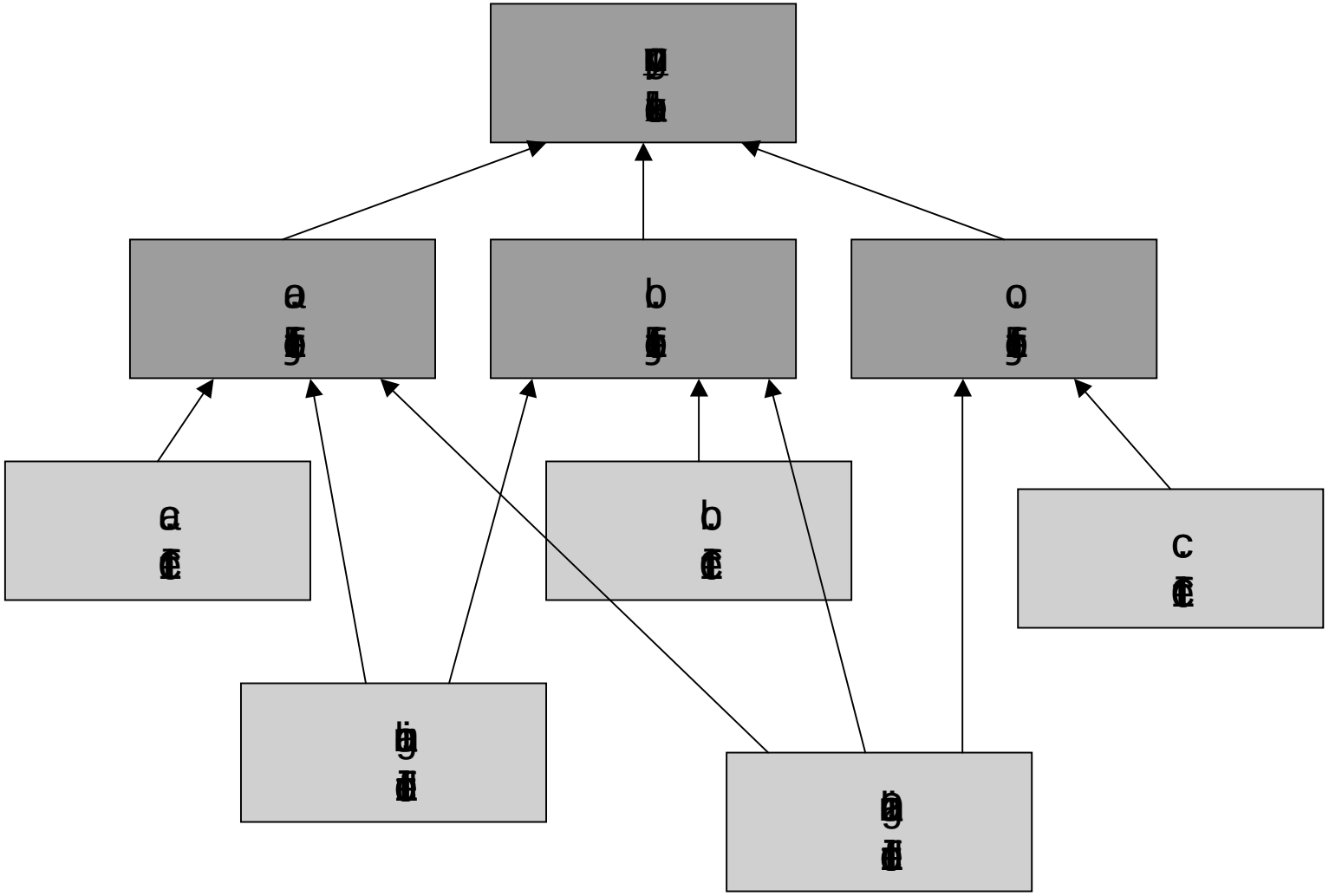
A simple [illegible]

[illegible]

Dependencies



Dependencies



Revision Control Systems

- Archival system for versions of files
  - File versions may be “checked in”, and retrieved later if necessary
  - Helps protects against accidental deletion, ill conceived editing etc.
  - Enables old versions of a project to be reconstructed
  - Also provides rudimentary “file locking” to assist in projects with multiple programmers

Checking files in/out

- Best to create and RCS directory
- ci
  - Checks file in to database, drops the lock
  - Enter description of the file
    - Further ci's will request a log entry
- co
  - Defaults to checking out the latest revision
  - Doesn't grab the lock, hence file is read-only

Locking files for editing

- co -l
  - Takes the lock (if available!) & leaves a read/write copy for editing
- ci -l
  - Checks a version in, but retains the lock
- Can view a prior version with:
  - co -p
- View rcs log with rlog
- Auto-annotate source files with Id

Other useful tools

- View differences between versions
  - diff
    - Defaults to current version
- Create a ‘symbolic name’ for a set of source files
  - tag
- See cvs or svn for more demanding revision control applications

Comparing files :df

- [illegible]
- c generate easier to read ‘context diff’
  - Show lines of context
- u GNU ‘unified diff’ : easiest to read & robust
- From-file / to-file can be directories
  - r recursive over sub-tree
  - N handle new/deleted files
- b ignore whitespace changes
- i ignore case distinctions

‘Packaging’ files : tar

- Creates an ‘archive’ containing sub-tree
- Preserves file system structure
- Typical usage:
  - create: tar -cf
  - extract: tar -xf
- GNU -z extension, equivalent to e.g.:
  - gzip -dc | tar -xf

[illegible]

- Takes output from diff to ‘patch’ a file/tree
- Patch files best created with:

  [illegible]

- Apply patch with:

  [illegible]

- [illegible] strips off the first pathname component
- Use -R to reverse patch

Data manipulation tools : tr

- is used to translate one character to another
- Character substitution:

  convert to lower case:

  “encrypt”:

- Deleting particular characters, e.g. CR, Ctrl-Z

  (note use of octal to specify Ctrl-M & Ctrl-Z)

- Translate and “squeeze repeats”

  Convert all numeric strings to a single ‘#’:

  Removing repeated spaces, tabs and NL’s:

Data manipulation tools : sort

- Sort lines in files based on key field list (primary 1st)
- By default, fields delimited by whitespace
  - Specify separator with: -t e.g. -t:
- By default, sort key is entire line
  - Key fields specified with: -k
  - E.g. -k2,4 specifies fields 2 through 4
- Treat field as numeric with -n
- Reverse sort order with -r
- Remove duplicate lines in sorted output with -u

Data manipulation tools : cut

- Extracts sections from lines in its input
- Field extraction:

  cut -f

  - Multiple fields (can't change order)

    cut -f
- Extract by character column:

  cut -c

  [illegible]

Data manipulation tools : sed

- Typ. used for regexp search/replace and delete
- Takes options on cmd line or in file with:

  [illegible]
- Search/replace (Substitute Global)

  [illegible]
  - Without 'g' only 1st match on each line is replaced
- Commands can preceded with line addresses:

  [illegible]
  - $ is last line of file

Data manipulation tools : sed

- Delete command : d
- Addresses can be regexps too! E.g.
  - Delete lines containing match

    /match/d
  - Substitute lines defined by region

    /start/,/end/d
- Text variables:
  - Use \(\) to capture strings into vars \1 \2
  - E.g. re-order 3 space-separated columns into 3 1 2

    s/\(.*\) \(.*\) \(.*\)/\3 \1 \2/

Selecting the right tool for the job

- Other scripting languages offer a more integrated & efficient environment than shell scripts
- [illegible] and GNU [illegible]
  - Good for data manipulation & processing
- [illegible]
  - At least as powerful as C/C++
    - Integrates well with other programming languages
    - Many high-level features and add-on modules
  - Auto-compiled to interpreted byte code
    - Respectable performance

[illegible]

- Use info pages to find out more
- Simple one-liners:

[illegible]

- Execute commands in a script with [illegible]
  - Also, [illegible] mechanism to create executable scripts

kscripts

- gsub substitute global (old, new)
- FNR number of records read in this file
- NF number of fields in current record
- $(i-1) using an index to access fields

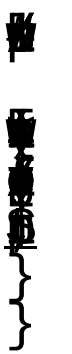
scripts

- !~ 'does not match' operator
- arr[x,y] 2D sparse array with auto-initialisation
- >filename file creation and redirection
- arr[x,y]+0 fixing the type of initialised elements

[illegible]

- Compact representation…
- …but, lots of cryptic operators, significant punctuation, strange default behaviour etc…
- However, it is widely installed and used
- Most scripts are CGI scripts run on web servers

scripts



- option enables some warnings (recommended!)
- angle operator reads next line of input to
- Regexps enclosed in
- Curly braces following if are not optional
- could be abbreviated to just
- Run scripts on the command line with
  - Iterate over all line is input with

arrays & associative arrays

- chomp removes at ends of lines
- split creates an array of whitespace separated words
- denotes entire array (# elts in array in a scalar context)
- denotes an element of an associative array
- denotes entire associative array

‘default behaviour’

- returns an array of strings separated by ‘:’
  - Defaults to operating on $_
  - Call to split has one argument, hence no need for brackets
  - 1st elt of array is [0] (not 1!)
- Line2 assigns the strip result array to the array
- is a shorthand for whitespace characters (not ws is)
  - One on many

More

```
[illegible]
```

- Use of back quotes to exec command and capture output to scalar (could have used an array for multi line output)
- =~ is the match operator ($_ unaltered in this case)
- Note use of brackets for grouping and field capture
- \s matches whitespace chars, \d decimal digits
- Note use of s? To match *day* and *days*

Python

- A little more verbose than Perl, but arguably more readable and less prone to bugs
- Clean object-oriented design
- Many add-on classes available
- Whitespace is significant!
  - Uses indentation instead of curly braces

目录

第一章 绪论 1

第二章 文献综述 12

第三章 研究方法 25

第四章 结果分析 38

第五章 讨论 51

第六章 结论 63

参考文献 70

附录 75

致谢 80