CSE2031 Software Tools - UNIX scripting

Summer 2010

Przemyslaw Pawluk

Department of Computer Science and Engineering
York University
Toronto

July 13, 2010

Notes

Table of contents

Notes

Shell and commands

Shell

Shell is the program that interprets your requests to run programs

Command

- single word like i.e. `who` or command plus args
- ends with `newline` or ;
- & runs command in background

Notes

Shell meta-characters

Notes

Variables

Environment variables

`$HOME, $PATH, $USER, $MAIL, $PWD, $HOST,`

Defined by user

No need to declare. Accessed by $

Notes

Cammand processing

- Variable substitution
- Command execution by shell
  - Builtin commands executed within shell process (Example: cd)
  - For all other commands the shell
    - Scans search path for file with same name
    - Uses fork() to create a new process
    - Uses exec() to load the program and execute it

Notes

Finding program

Search paths are storen in an variable `$PATH`. It is a list of directories separated by `':'`.These directories are searched in the order they are given.
Directories called "bin" typically contain programs under Unix
If a command name contains a `'/'`, the search path is not used i.e. `/usr/bin/cal`

Notes

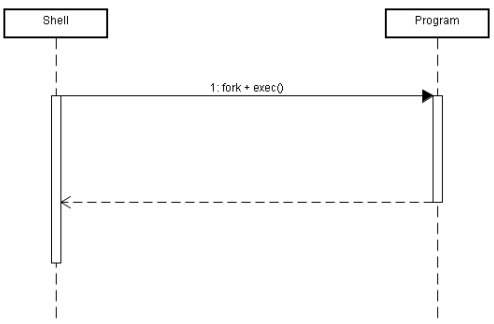
Program Execution

When executing a program, the shell:

- Starts a new process
- Executes the program file that it found
- Then waits for the program to finish

Notes

Shell calls and waits



Notes

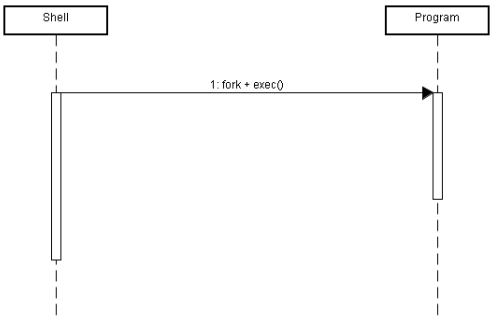
Background Tasks

- You can use the '&' character to tell the shell to run the program in the background
- A background process is one that is running but that you are not waiting for

```
echo $HOME &
```


Notes

Shell calls and does not wait



Notes

Wait

We can change our mind The command `wait` will make the shell stop until all background processes have finished

Notes

Conditional execution

```
command1 && command2
```

run command1; only if successful, run command2

```
command1 || command2
```

run command1; only if unsuccessful, run command2

Notes

Exit status

Every program in Unix has an exit status:

- 0 = success/true
- nonzero = failure/false

Note that this is opposite to C!

Notes

Loops

sh

```
for variable in list_of_values
do
    command1
    command2
        ...
done
```

bash loops (C-like)

```
for ((i=1; i <= 10 ; i++))
do
    commands
done
```


Notes

If-Else Statements

```
1  if (condition_command) then
2      command1
3      command2
4      ...
5      last_command
6  else
7      command1
8      command2
9      ...
10     last_command
11 fi
```


Notes

Quotes

Single

Single quotes ’ ’ – All characters inside single quotes are treated as nonspecial (except ’)

Back-quotes

Back-quotes ‘ ‘ – the contents of the quote is treated as a shell command but special characters are not processed

Double

Double-quotes " " like single quote except the variable substitution $ and backquotes ‘ are still treated as special characters

Notes

Comments

Comments start with a ’#’ character, terminated by newline

```
#this is a comment
```

In the first line of a shell program `#!<shell>` specifies which shell is used to run the script

```
#!/cs/local/bin/sh
```


Notes

Sub-shells

When needed shell calls fork to create new process (like fork() in C)

- We can do this explicitly with ( ) operator
- Causes command to be executed in a subshell
- Changes in subshell do not affect its parent!

Notes

Notes

Notes