**8**

**More on Commands**

**Objectives**

· Identify the order of substitution and action as the Shell interprets the command line

· Identify and set delimiters to group commands

·Explain Shell command search procedure



Notes

UNIX Shell Programming

©Copyright 1996-2000 GTRC,All Rights Reserved



More on Commands

**Command Line Evaluation**

**·When the Shell reads a command line, the** followingsubstitutionsand **actions occur before the command is** executed.

**1. read line**

**2. variable and parameter substitution**

**3. command substitution**

**4. file name generation**

**5. delimiter interpretation**

**6. I/O redirection**

<

**7. assignment of variable**

**8. search path**

**9. execute command**

**Notes**

$ 1s > allfi1es

Example:

What files are in allfiles?

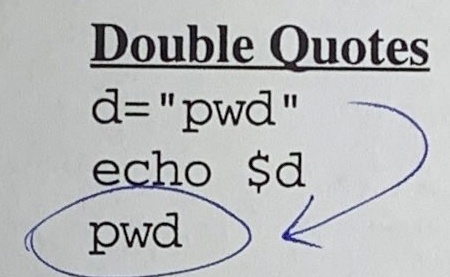
$ 1s \*> allfiles

The shell does file name generation before I/O redirection.

8-3

UNIX Shell Programming

©Copyright 1996-2000 GTRC,All Rights Reserved



More on Commands

**Command Line Evaluation (Cont'd)**

**1. Read Line**

**Quoted new line**

echo "Mary Smith

Atlanta, GA"

**2. Variable and Parameter Substitution**

**3. Command Substitution**

d=`pwd`

**4. File Name Generation**

**5. Delimiter Interpretation**

**6. I/O Redirection**

**7. Assignment of Variable**

**8. Search Paths**

**9. Execute Command**

**Notes**

1. The Shell reads and parses the command line up to the RETURN unless the RETURN is quoted.

2. (see chapter 5 and 6)

3. The output of a command can be assigned to a variable by enclosing the command in grave quotes.

Double quotes around a command will assign that command to the variable.

What is the difference between the two echo commands?

The double quoted command aliases itselfto the variable.

The grave quoted command is actually executed at the time it is assigned.

UNIX Shell Programming

8-4

© Copyright 1996-2000 GTRC,All Rights Reserved

4. See chapter 7

5. Blanks,tab,or newlines are the default characters of the string "$IFS". The command is broken into distinct arguments using the IFS delimiter.

6. Redirection occurs from left to right. (see chapter 3)

7. See chapter 5

8. If a command starts with a /,,/, or../,then the full pathname is used and no search is made.Using the full pathname gives a faster response.

The command search path can be altered by setting the PATH variable.

PATH=/bin:/usr/bin::

The::represents the current directory.

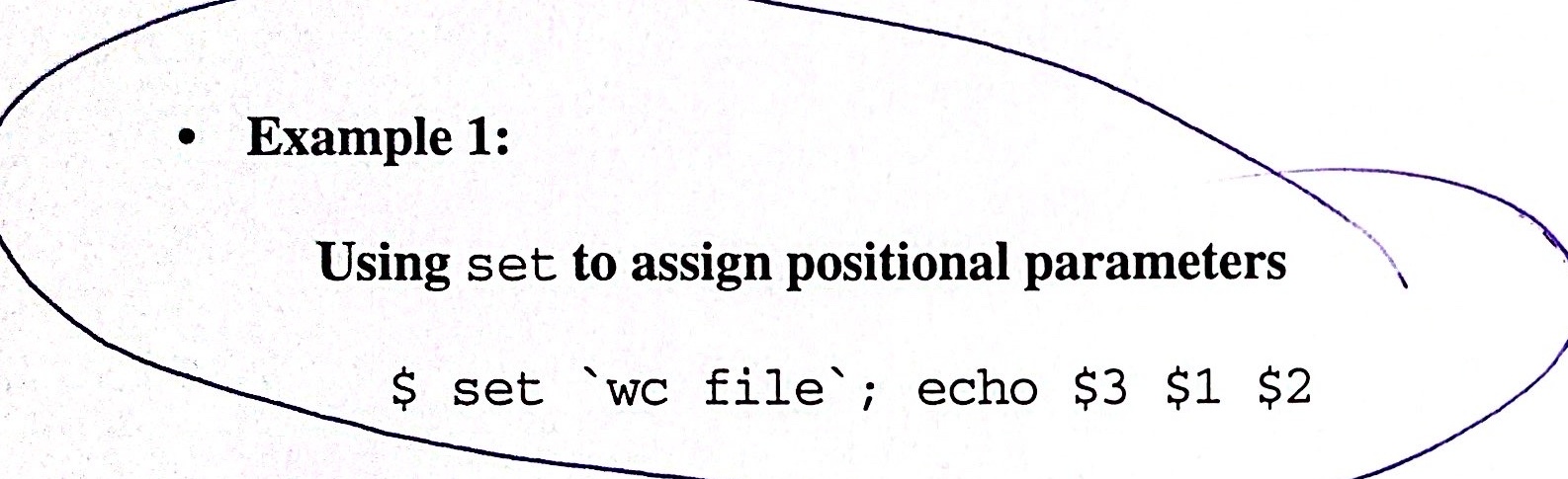
Error Messages:

If the command does not exist in one of the search paths, the message "not found"is displayed. If the command is found, it will execute the first file found by that name.

If the user does not have execute permission, the message "cannot execute” is displayed.

9. The Shell executes the command and returns the appropriate exit status.

**Command Substitution Examples**



**Example 2:**

**Command substitution can be nested.**

$ message="Today is `date'"

$ echo $message

**·Example 3:**

**You have a list of people you want to send a** memo **to in a file called** maillist.

$ mail `cat maillist < memo

**Notes**

Exercise:

Create the following file and produce the output.

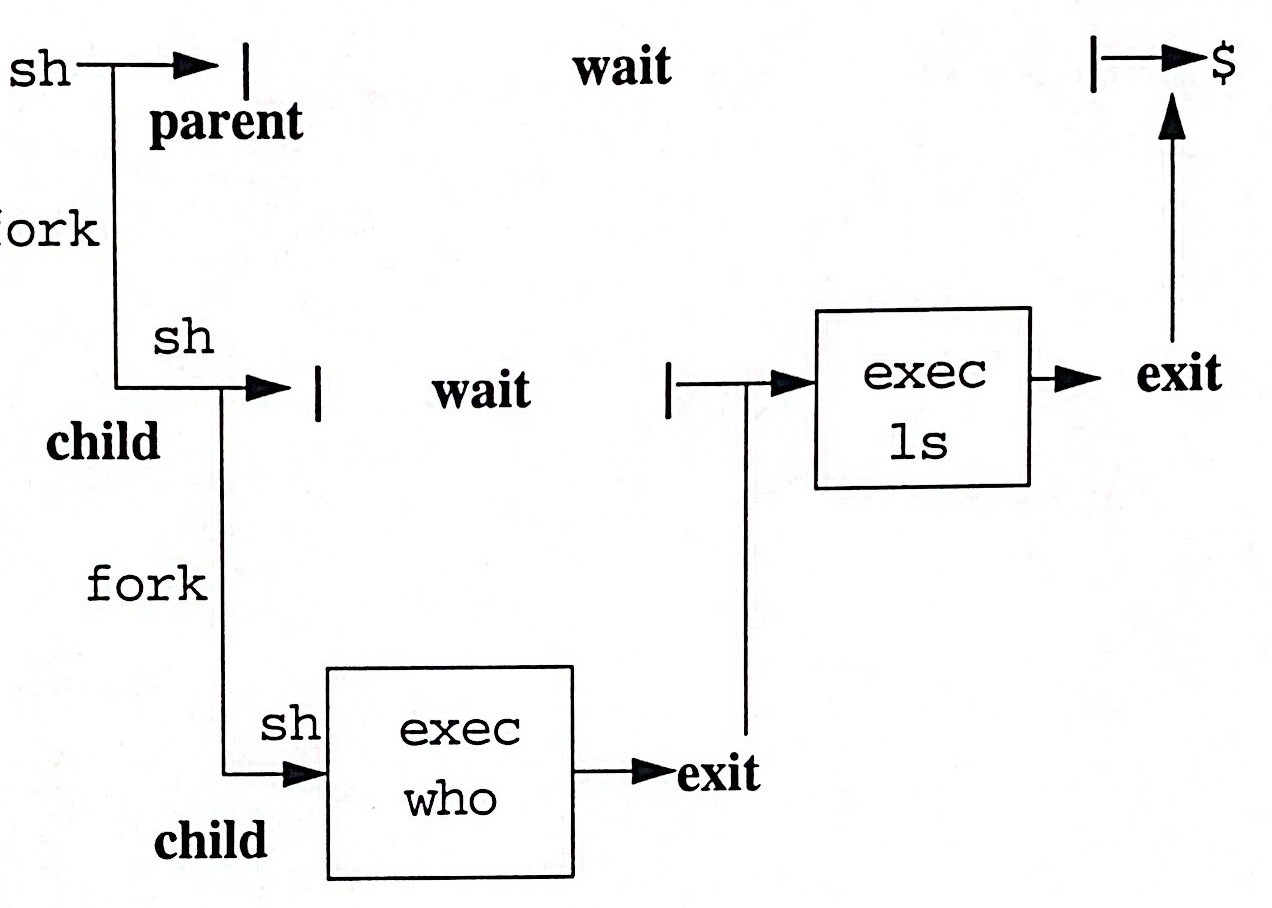
echo no quotes: term is $TERM, files are \*

echo single quotes: 'term is $TERM, files are \*

echo double quotes: "term is $TERM, files are \*"

8-6

**Shell Script**

$ (who;1s)

**process** **1**

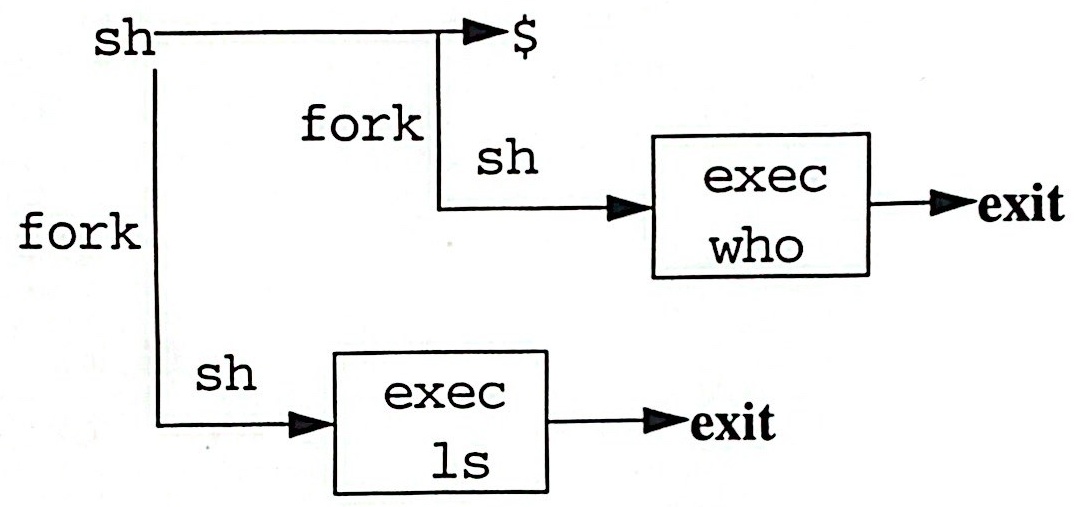
**process 2**

process 3

Notes

**Shell Script (Cont'd)**

$ who&ls&

process 1

process 3

process 2



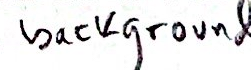
Notes

**Command Delimiter Examples**

**1. The Shell reads up to a new line and executes.You can put many commands** **on one line, but you have to use a semicolon to distinguish between** **commands.**

$ echo hello; echo How are you?; date

**2.Using semicolons as separators, causes the commands to be executed** **sequentially. & can also be used as a command delimiter.**



$ (date; who)& echo done

**3. It is not known which process,** date **or** echo, **will start first.who will not** **start until** date **finishes.**

$ (date; who)& echo done&

**Notes**

Example:

Create a group of executable files a,b,and c.

$ cat a

echo 'aaaaa...a'

echo 'aaaaa...a'

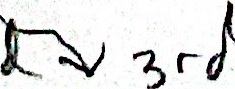
Execute the following commands:

$ (a;b)&c



$a& b& c&

$ (a;b;c)&



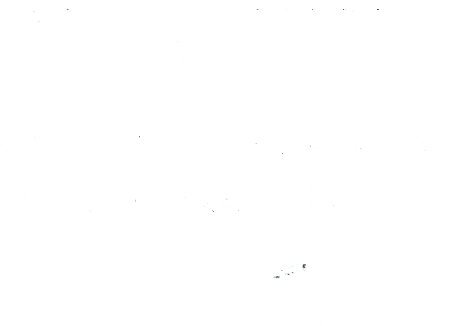
UNIX Shell Programming 8-9

© Copyright 1996-2000 GTRC, All Rights Reserved

**Short-Circuit Operators**

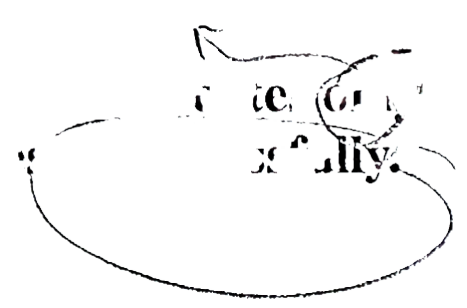
**&& and|**

**The "and" and "or" operators are short-circuit operators.Evaluation is-made left to right and terminates as soon as** **truth or falsehood is determined.** 

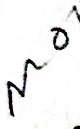
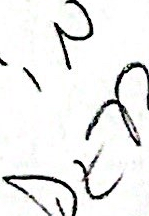
&& command1 && command2

command2 **executes only if the first command** command1**executes successfully.**

command1 command2

command2 **executes only if the first command** command1

**executes unsuccessfully.**



**Notes**

·Start-up database or help me.

isql || echo "problem in starting database"

true -> no message

false -> message

**Braces and Parentheses**

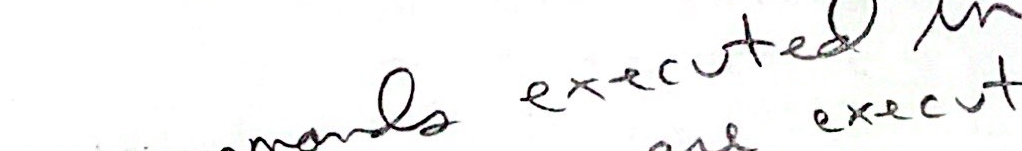
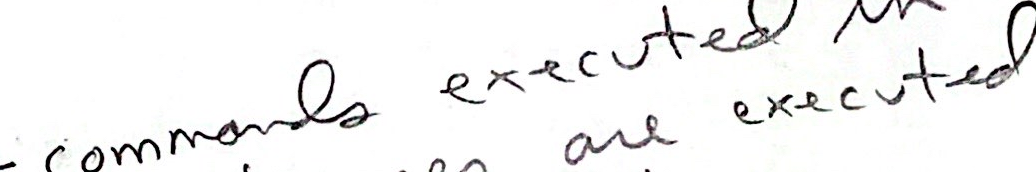
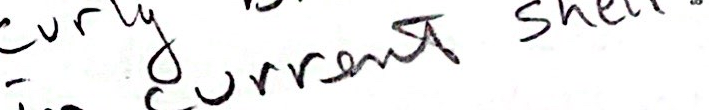
**There are two ways of grouping commands:**

**1.Use braces** {}

$ cd

$ pwd

$HOME/team1

$ { cd /usr/bin; 1s;

<files in /usr/bin>

$ pwd

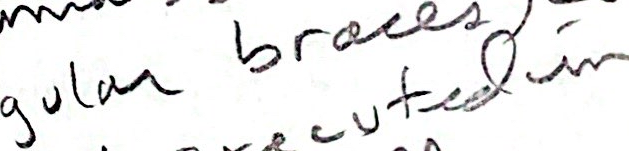
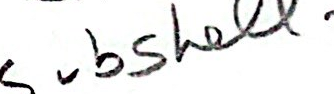
/usr/bin

**2. Use parentheses** ()

$ cd

$ pwd

$HOME/team1

Co  regulan braces  

$ (cd /usr/bin; 1s)

<files in /usr/bin>

$ pwd

SHOME/team1

**Notes**

What is the difference in the two sequences of commands?

In the first sequence, the command list between the braces is simply executed. In the second sequence, the command list between the parentheses is executed as a separate process.

**Summary**

Commands are evaluated in the following order:

1. Line is read

2. Variable and parameter substitution

3. Command substitution

4. File name generation

5. Delimiter interpretation

6. I/O redirection

7. Variable assignments

8. Path search

9. Command execution

& and ; delimit commands. & puts commands in the background.

 ackgrounde 

&& and | | also delimit commands. && is the logical AND operator.The second command is executed only if the first succeeded.| | is the logical OR operator. The second command is executed only if the first one fails.

Braces delimit commands that are executed in the current shell.

Parentheses delimit commands that are executed in a subshell.



Notes



**Lab 8**

**Objective:**

·You will confirm the order of file name generation versus I/O redirection. Command generation can be part of the command line and the output of commands can be assigned to variables. You will test these concepts and variables. You will use parentheses and braces to alter the execution of commands and creation of subshells.

**Exercises:**

1. Construct an example to illustrate that file name generation occurs before I/O redirection.(Hint: use 1s)

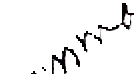
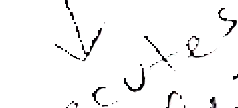
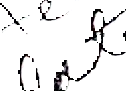
2. Execute the following sequence of commands and determine the difference.

Grave Quotes Double Quotes

d='date' 

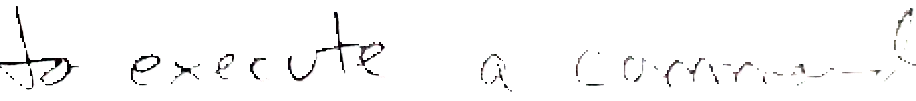
echo $d echo $d

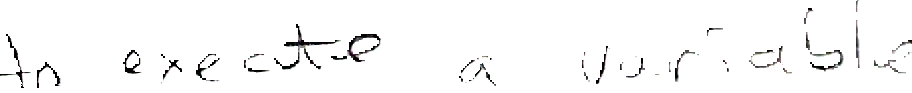
date



<displays date>

When would you use grave quotes? 

When would you use double quotes? 

3. Execute the following sequence of commands and determine the difference

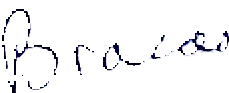
Parentheses **Braces**

echo $1 $2 echo $1 $2

(set 'date';echo $1 $2) { set 'date'; echo $1 $2;}

echo $1 $2 echo $1 $2

 are used to group output

 are used to create a subshell



UNIX Shell Programming 8-13

© Copyright 1996-2000 GTRC, All Rights Reserved

4. Create a mailing list (put your name in the list for verification purposes). Create a memo to be sent to everyone in the list. Using only one command line, send the memo to everyone in the list.

5. Create three executable files.The first file a echoes aa's, the second file b echoes bb's and the third file c echoes cc's. Each file should have at least 5 lines with 20 characters per line.Make each file executable. Perform the following commands:

(a;b)&c

(a;b)& c&

a& b& c&

6. Using the && and || operators, execute the following commands.

y=`who | wc -1

expr $y \> 3 && echo "$y is too many"

expr $y \> 3 || echo "logon"