Passing Arguments

The shift Command
Objectives

• Shell programs become far more useful after you learn how to process arguments passed to them.

• You’ll learn how to write shell programs that take arguments typed on the command line.
Positional Parameters

• Whenever you execute a shell program, the shell automatically stores the first argument in the special shell variable 1, the second argument in the variable 2, and so on.
• These special variables - more formally known as *positional parameters* - are assigned after the shell has done its normal command-line processing (that is, I/O redirection, variable substitution, filename substitution, and so on).
Passing Argument Example

• $ cat ison
  who | grep $1

• $ who
  barney   tty03   July 2  08:37
  fred    tty04   July 1  07:34

• $ ison zhang

• $ ison fred
  fred   tty04   July 1  07:34
The `#$` Variable

- Whenever you have a shell program executed, the special shell variable `#$` gets set to the number of arguments that were typed on the command line. For example,

  ```sh
  $ cat args
  echo #$ arguments passed
  echo arg 1 = :$1: arg 2 = :$2: arg 3 = :$3:
  ```
The `$#` Variable Example

- $ args a b c
  $ args a b
  $ args
  $ args “a b c”
  $ ls intro*
  $ args intro*
  $ my_bin = /home/jzhang/Unix_work/chap7
  $ args $my_bin
  $ args $(cat names)

- The shell does its normal command-line processing even when it’s executing your shell programs. This means that you can take advantage of the normal niceties such as filename substitution and variable substitution when specifying arguments to your programs.
The $ * Variable

• The special variable $ * references all the arguments passed to the program.

• This is useful in programs that take an indeterminate or variable number of arguments. For illustration,
$ cat args2
echo $# arguments passed
echo they are :$*:
The $ * $ Variable Example

- $\texttt{args2 a b c}$
- $\texttt{args2 one two}$
- $\texttt{args2}$
- $\texttt{args2 *}$
A Program to Look Up Someone in the Phone Book

- $ grep Cheb phonebook
  $ grep “Susan T” phonebook
- $ cat lu
  
  Look someone up in the phone book
  
  grep #1 phonebook
- $ lu Alice
  $ lu Susan
  $ lu “Susan T” (revise the #1 argument to grep)
- $ lu Tony
  $ lu “Susan T”
A Program to Add Someone to the Phone Book

- $ cat add
  
  #
  # Add someone to the phone book
  #

  echo “#1 (tab) #2” >> phonebook
- $ add ‘PapaJohn Pizza’ 973-234-2434
  $ lu Pizza
  $ cat phonebook
- sort -o phonebook phonebook
- $ add “Billy Bach’ 203-234-2354
A Program to Remove Someone from the Phone Book

• $ cat rem
  #
  # Remove someone from the phone book
  #

grep -v "#1" phonebook > /tmp/phonebook
mv /tmp/phonebook phonebook

• $ rem ‘PapaJohn Pizza’
  $ lu Pizza
  $ cat phonebook

• $ rem Susan
  $ cat phonebook
  $ add ‘Susan Goldberg’ 201-234-3242
  $ add ‘Susan Topple’ 234-435-2156

• sed “/$1/d” phonebook > /tmp/phonebook
If more than nine arguments to a program, you cannot access the tenth and greater arguments with $10, $11, and so on.

If you try to access the tenth argument by writing $10, the shell actually substitutes the value of $1 followed by a 0.

Instead, the format `${n}` must be used. For example, to directly access argument 10, you must write `${10}` in your program.