Defining Your Own Variables

1 Using Set Commands to Define Your Own Variables

- The syntax for the **set** command illustrated:
  `set var1=3`
  sets `var1` to the integer `3`;
  `set var1=stuff`
  sets `var1` to the string `'stuff'`.

  `set var0=This is a test`
  sets `var0` to `'This`';

  `set var0='This is a test'`
  sets `var0` to `'This is a test'`.
- To **refer** to a variable, use a `$` before the variable's name:

  `cp $var1 save.stuff`
  copies `stuff` to `save.stuff`.
- Variables can be used in set statements:

  `set var2=better$var1`
  sets `var2` to `'betterstuff'`.
- To **display** the value of all set variables, execute the command:

  `set`
  whereas
  `set var1`
  sets `var1` to null!
- To display a **single** set variable, say `var1`, execute:

```
echo $var1
```

2 Array Variables

- These variables also are known as **indexed** or **multivalued**.
- Initialization involves parentheses:

```
% set manyvalues = (1st 2nd 3rd 4th)
% echo $manyvalues
1st 2nd 3rd 4th
% echo $#manyvalues
4
% set manyvalues[2] = not2nd
% echo $manyvalues
1st not2nd 3rd 4th
% set test = '1st not2nd 3rd 4th'
% echo $#test
1
% set manyvalues = ($manyvalues 5th 6th)
% echo $manyvalues
1st not2nd 3rd 4th 5th 6th
```

- The path variable is an array.

```
% echo $path
. /usr/local/bin /usr/ucb /bin /usr/bin /fs/
hydra2/home/cappello/Apps /LocalApps /NextApps .
/fs/hydra2/home/cappello/l /fs/hydra2/home/cap-
pello/b /LocalApps/fmbib/bin /usr/ucb /bin /usr/
lib/tex/macros /usr/local/tex/bin /usr/bin /usr/
local /usr/compsce/tex/bin /usr/compsci/bin /usr/
local/bin /eci/rn/bin
```

- The shift command deletes the 1st element of an array.

```
% shift manyvalues
% echo $manyvalues
not2nd 3rd 4th 5th 6th
```

- Variables defined by the `set` command are **private** to the C shell that created them.

3 Using the @ operator to Define a Variable

The @ operator can be used to define a set variable that contains an expression.

```
% @ j = 10 * 5
% echo $j
50
% @ k = $j / 2
% echo $k
25
```

4 Displaying the Current Value of a Shell Variable

```
% set ugh = stuff
% echo $ugh
stuff
```

- When a variable is in **double** quotes, shell variables are substituted for:

```
% echo "$ugh"
stuff
```

- When a variable is in **single** quotes, shell variables are **not** substituted for:

```
% echo '$ugh'
$ugh
```

- The placement of a \ also prevents substitution:

```
% echo \$ugh
$ugh
```

5 Deleting a Shell Variable

- unset destroys a shell variable:

```
% set stuff
% echo $stuff
%
% unset stuff
% echo $stuff
stuff: Undefined variable.
```

6 Environment Variables

- An environment variable has all the properties of a set variable with one addition: the definition of an environment variable is passed to any C shell that is started by the C shell in which it was defined.

```
% setenv WORK ~cappello/temp
% printenv WORK
/fs/hydra2/home/cappello/temp
% echo $WORK
/fs/hydra2/home/cappello/temp
```

- It is customay to use purely UPPER case letters to distinguish `set` variables from `environment` variables.
- To illustrate the block-structured nature of environment variables:

```
% set p = goodstuff
% setenv W badstuff
% csh
% echo $p
p: Undefined variable.
% echo $W
badstuff
```

7 Determining Whether a Variable Is Defined

- The expression `$?variable` evaluates to true if and only if the variable `'variable'` is defined.

```
if ($?WORK) then
  echo "WORK is defined to be $WORK"
else
  echo "WORK is not defined"
endif
```

More about conditional statements later.

- There currently is no way to test if a variable is a shell variable or an environment variable.

  Enhancements to the C shell may change this.

8 Deleting an Environment Variable

```
unsetenv WORK
```

undefines the environment variable `WORK`.

- This command accepts **metacharacters** (as does the **unset** command):

```
unsetenv WORK*
```

  undefines all environment variables that have `WORK` as a prefix.

  (More about metacharacters later.)

9 Controlling Variable Substitution

- Braces `{ }` can be used to delimit the name of a variable in an expression:

-

```
% set dir1 = ~cappello/i/2
% ls $dir12
dir12: Undefined variable.
% ls ${dir1}2
Admin                C                Exams
Homework             Notes
%
```

- **Command aliasing precedes variable substitution!**