OS Programming

Shell Script Programming

Objectives

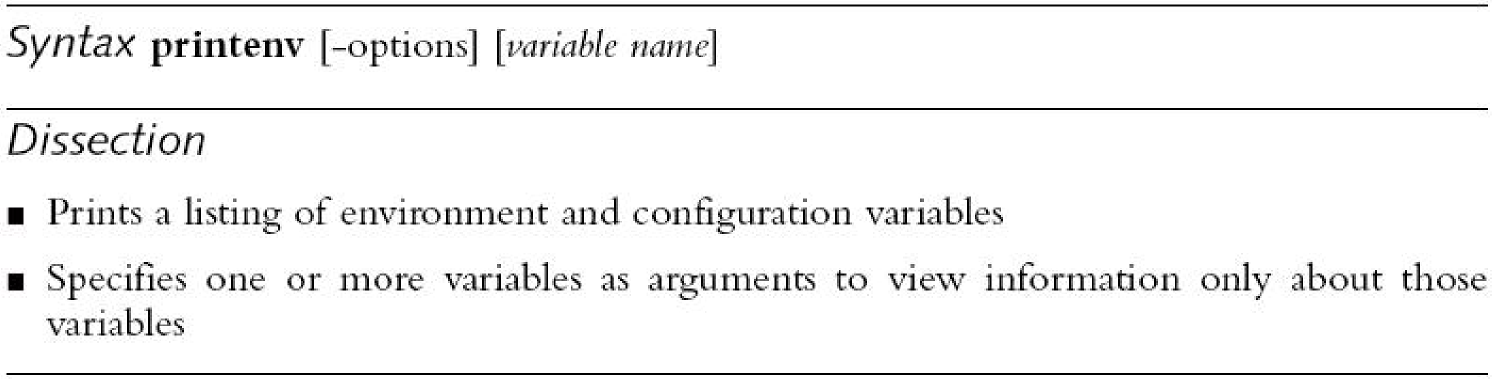
- See different types of variables
- Learn to set environment and assign shell variables
- Write interactive shell scripts

Types of Variables

- **Configuration variables**
  - Store information about the setup of OS
  - Not typically modified after they are set up
- **Environment variables**
  - Initial values can be changed as needed
- **Shell variables** are created at command line or in a shell script
  - Useful for temporarily storing information

Environment and Configuration Variables (continued)

- Use *printenv* to view list of your current environment and configuration variables



Syntax **printenv** [-options] [*variable name*]

Dissection

- Prints a listing of environment and configuration variables
- Specifies one or more variables as arguments to view information only about those variables

- Use *set* (no arguments) to view current Bash shell environment
  - Including environment variables, shell script variables, and shell functions

Table 6-2 Standard Bash shell environment and configuration variables

Name	Variable Contents	Determined by
HOME	Identifies the path name for user's home directory	System
LOGNAME	Holds the account name of the user currently logged in	System
PPID	Refers to the parent ID of the shell	System
TZ	Holds the time zone set for use by the system	System
IFS	Enables the user to specify a default delimiter for use in working with files	Redefinable
LINEND	Holds the current line number of a function or script	Redefinable
MAIL	Identifies the name of the mail file checked by the mail utility for received messages	Redefinable
MAILCHECK	Identifies the interval for checking and received mail (example: 60)	Redefinable
PATH	Holds the list of path names for directories searched for executable commands	Redefinable
PS1	Holds the primary shell prompt	Redefinable
PS2	Contains the secondary shell prompt	Redefinable
PS3 and PS4	Holds prompts used by the *set* and *select* commands	Redefinable
SHELL	Holds the path name of the program for the type of shell you are using	Redefinable
BASH	Contains the absolute path to the Bash shell, such as /bin/bash	User defined
BASH_VERSION	Holds the version number of Bash	User defined
CDPATH	Identifies the path names for directories searched by the *cd* command for subdirectories	User defined
ENV	Contains the file name containing commands to initialize the shell, as in .bashrc or .tcshrc	User defined
EUID	Holds the user identification number (UID) of the currently logged in user	User defined
EXINIT	Contains the initialization commands for the vi editor	User defined
FCEDIT	Enables you to access a range of commands in the command history file; FCEDIT is a Bash shell utility and is the variable used to specify which editor (vi by default) is used when you invoke the FC command	User defined
FIGNORE	Specifies file name suffixes to ignore when working with certain files	User defined

Table 6-2 Standard Bash shell environment and configuration variables (continued)

Name	Variable Contents	Determined by
FUNCNAME	Contains the name of the function that is running, or is empty if there is no shell function running	User defined
GROUPS	Identifies the current user's group memberships	User defined
HISTCMD	Contains the sequence number that the currently active command is assigned in the history index of commands that already have been used	User defined
HISTFILE	Identifies the file in which the history of the previously executed commands is stored	User defined
HISTFILESIZE	Sets the upward limit of command lines that can be stored in the file specified by the HISTFILE variable	User defined
HISTSIZE	Establishes the upward limit of commands that the Bash shell can recall	User defined
HOSTFILE	Holds the name of the file that provides the Bash shell with information about its network host name (such as *localhost.localdomain*) and IP address (such as *129.0.0.24*); if the HOSTFILE variable is empty, the system uses the file /etc/hosts by default	User defined
HOSTTYPE	Contains information about the type of computer that is hosting the Bash shell, such as i386 for an Intel-based processor	User defined
INPUTRC	Identifies the file name for the Readline start-up file overriding the default of /etc/inputrc	User defined
MACHTYPE	Identifies the type of system, including CPU, operating system, and desktop	User defined
MAILPATH	Contains a list of mail files to be checked by mail for received messages	User defined
MAILWARNING	Enables (when set) the user to determine if she has already read the mail currently in the mail file	User defined
OLDPWD	Identifies the directory accessed just before the current directory	User defined
OPTIND	Shows the index number of the argument to be processed next, when a command is run using one or more option arguments	User defined
OPTARG	Contains the last option specified when a command is run using one or more option arguments	User defined

Table 6-2 Standard Bash shell environment and configuration variables (continued)

Name	Variable Contents	Determined by
OPTERR	Enables Bash to display error messages associated with command-option arguments, if set to 1 (which is the default established each time the Bash shell is invoked)	User defined
OSTYPE	Identifies the type of operating system on which Bash is running, such as linux-gnu	User defined
PROMPT_ COMMAND	Holds the command to be executed prior to displaying a primary prompt	User defined
PWD	Holds the name of the directory that is currently accessed	User defined
RANDOM	Yields a random integer each time it is called, but you must first assign a value to the RANDOM variable to properly initialize random number generation	User defined
REPLY	Specifies the line to read as input, when there is no input argument passed to the built-in shell command, which is read	User defined
SHLVL	Contains the number of times Bash is invoked plus one, such as the value 3 when there are two Bash (terminal) sessions currently running	User defined
TERM	Contains the name of the terminal type in use by the Bash shell	User defined
TIMEFORMAT	Contains the timing for pipelines	User defined
TMOUT	Enables Bash to stop or close due to inactivity at the command prompt, after waiting the number of seconds specified in the TMOUT variable (TMOUT is empty by default so that Bash does not automatically stop due to inactivity.)	User defined
UID	Holds the user identification number of the currently logged in user	User defined

Modifying the PATH Variable

- The shell looks for programs in the PATH
  - *./filename* runs script
    - ./ needed if current directory is not in PATH
- To see the directories in your path:

```
echo $PATH
```

  - Sample output:

```
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin
```

- To add the current working directory to the PATH:

```
PATH=$PATH:.
```

Shell Variables

Create a variable ***msg*** and assign it to ***"hello"***

```
msg=hello
```

You cannot have space around the '=' operator. I.e., do not do this `msg = hello`

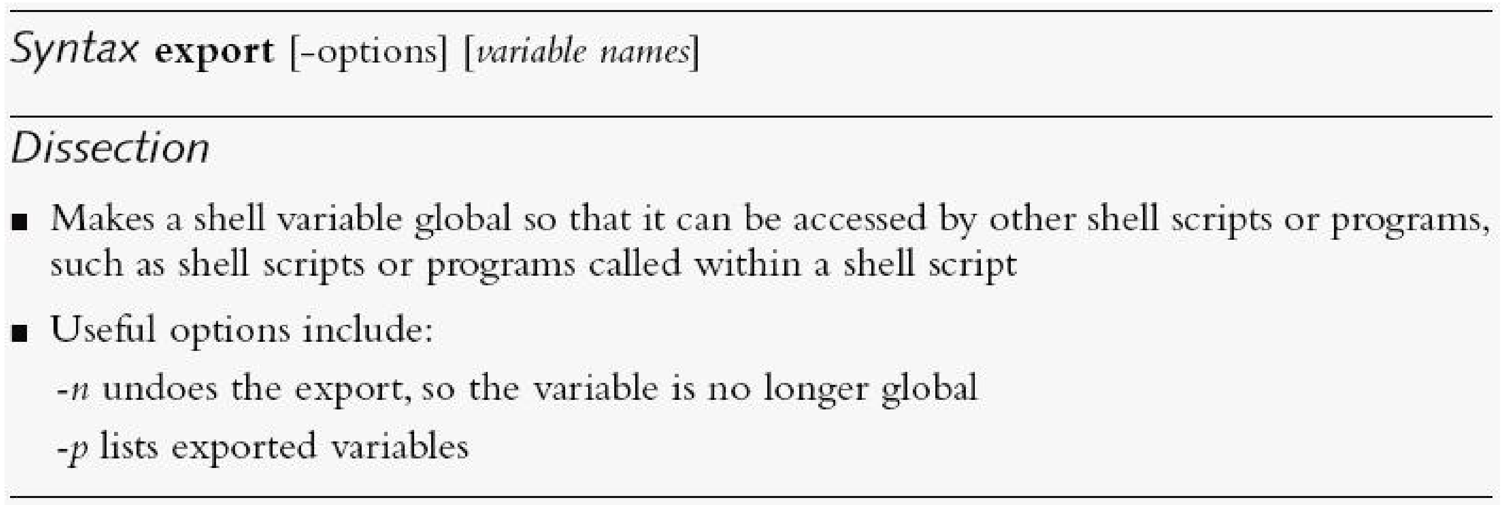
Pring the variable

```
echo $msg
```

Bash variables contain strings. There are no other types. Some commands interpret strings as numbers, etc.

Exporting Shell Variables to the Environment

- Scripts cannot automatically access variables created/assigned on command line or by other scripts
  - You must use *export* first



Syntax **export** [-options] [*variable names*]

Dissection

- Makes a shell variable global so that it can be accessed by other shell scripts or programs, such as shell scripts or programs called within a shell script
- Useful options include:

  -*n* undoes the export, so the variable is no longer global

  -*p* lists exported variables

Defining Operators

- **Defining operators:** assigns a value to a variable
  - Examples:

```
NAME=Becky
NAME="Becky J. Zubrow"
LIST=`ls`
LIST=$(ls)
```

Evaluating Operators

- Display contents of a variable via an **evaluating operator**
  - Examples:

```
echo $NAME
echo "$NAME"
echo '$NAME'
echo `$NAME`
```

Be aware of what type of quote to use.

Examples:

```
echo echo $NAME
echo "echo $NAME"
echo 'echo $NAME'
echo `echo $NAME`
```