Chapter 10

Functions

Overview

- Shell Functions
- Passing Arguments to Functions
- Returning Values from Functions
- Function Libraries

Lesson: Shell Functions

- Syntax:

```
fname() compound-command [redirection ...]
function fname { compound-command ; }
```

  - Defines named compound command (can be list)
- Functions run like . -scripts, in caller's context.
- Functions run when called, not when defined.
- Functions will not be exported to subshells.
- Example:

```
day() date +'%A, %B %e'
function holder {
  echo "\nPRESS RETURN TO CONTINUE"
  read anything
}
```

Displaying Current Shell Functions

- To display the contents of declared functions:

```
$ typeset -f
function function_name
{
    command_lines
}
day() date +'%A, %B %e‘
```

- To display the names of declared functions:

```
$ typeset -F
function_name1
function_name2
```

Declaring Functions in a Shell Script

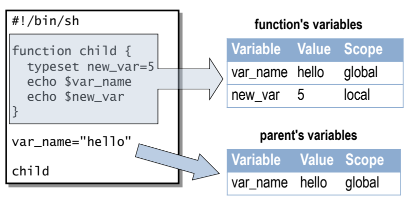
- It is good practice to declare all functions at the start of the script file.
- You can also dot-source them from another file.

```
$ cat funclib
func1() ...
func2() ...

$ cat myscript
#!/bin/sh -x
. ~/funclib
func3() {
  ...
}
...
commands
...
```

Local Variables

- Korn shell family shells can define local variables in function



```
#!/bin/sh

function child {
  typeset new_var=5
  echo $var_name
  echo $new_var
}

var_name="hello"

child
```

function's variables

Variable	Value	Scope
var_name	hello	global
new_var	5	local

parent's variables

Variable	Value	Scope
var_name	hello	global

Lesson: Passing Arguments to Functions

- **Functions have their own positional parameters.**
- **You can pass parameters to function when calling it:**

```
shifter() {
  echo "$# parameters passed to $0"
  while [ $# -gt 0 ]
  do
    echo "$*"
    shift
  done
}
             # MAIN BODY OF THE SCRIPT
echo "Please type a list of five words: "
read varlist
set $varlist # creates positional parameters
shifter $*   # pass arguments to function
echo "$# parameters in the parent"
echo "Parameters: $*"
```

Lesson: Returning Values from Functions

- **The `return` Command**
- **Function Output**

The `return` Command

- **Syntax:**

```
return [n]
```

  - Stops executing current function or dot-script and returns *n* as exit code.
- **Example:**

```
$ sqr1() { return $(($1 * $1)); }
sqr1 5
echo $?
25
```

- **Exit code is 1-byte number, so only small integers (less than 256) can be returned**

Function Output

- Function output will go to `stdout` and can be redirected, piped or captured (command substitution)

```
#!/bin/sh
function child {
   typeset new_var=5
   echo $var_name
   echo $new_var
}
var_name="hello"
fun_data=$(child)
echo "$fun_data"
```

function's variables

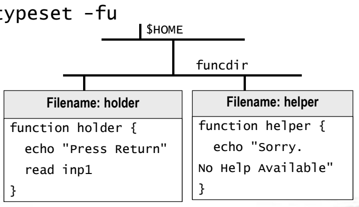
Variable	Value	Scope
var_name	hello	global
new_var	5	local

parent's variables

Variable	Value	Scope
var_name	hello	global
fun_data	hello 5	global

Lesson: Function Libraries

- Use dot-sourcing
- Korn shell family can use function autoloading
  - Define `FPATH` environment variable
  - Use `typeset -fu`



Filename: holder

```
function holder {
  echo "Press Return"
  read inp1
}
```

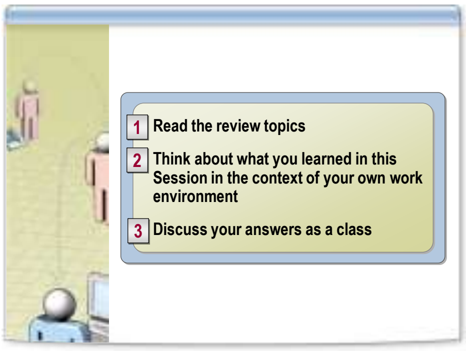
Filename: helper

```
function helper {
  echo "Sorry.
No Help Available"
}
```

Review Exercises

- Complete the exercises from the Learning Guide

Topics for Review



1. Read the review topics
2. Think about what you learned in this Session in the context of your own work environment
3. Discuss your answers as a class