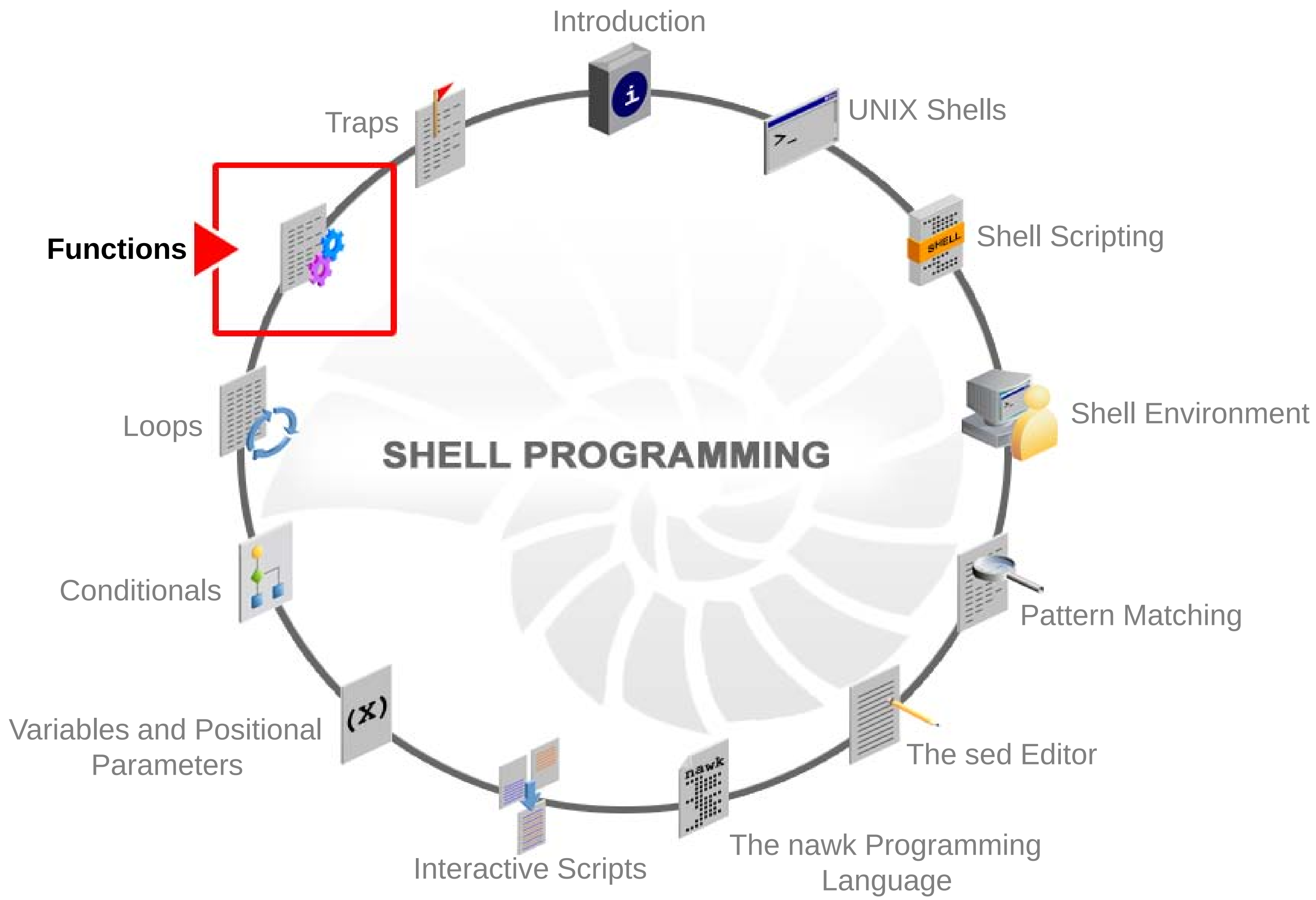
12

Functions

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Create user-defined functions in a shell script
- Use the `typeset` and `unset` statements in a function
- Autoload a function file into a shell script

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Agenda

- Creating user-defined functions in a shell script
- Using the `typeset` and `unset` statements in a function
- Autoloading a function file into a shell script

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Functions in a Shell

- A function is a set of one or more statements that act as a complete routine.
- Each function must have a unique name within a shell or shell script.
- Syntax:

```
function function_name [ block_of_statement_lines]
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Functions in a Shell: Example

```
#!/bin/bash

# Define your function here
Hello () {
   echo "Hello World"
}

# Invoke your function
Hello
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Positional Parameters and Functions

- Functions act like miniscripts, in that they can:
  - Accept parameters that are passed to them
  - Use local variables
  - Return values back to the calling shell command line
- Positional parameters passed to functions are not the same positional parameters that are passed to a script.
- The examples in the following slides illustrate that difference.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Positional Parameters and Functions

```
cat funparas.sh
#!/bin/bash
# Script name: funparas.sh
function hello
{
      print ’$1 the function is: ’ $1
}
print ’Input passed and stored in $1 is: ’ $1

hello John   # execute the function hello

print
print ’After the function $1 is still ’ $1

$ ./funparas.sh Susan
Input passed and stored in $1 is: Susan
$1 in the function is: John
After the function $1 is still Susan
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Positional Parameters and Functions

```
$ cat ~/.bashrc

killit ()    # Bash shell syntax
{
  pkill -u $1
}


function rcgrep    # Bash Shell syntax
{
  grep $1 /etc/init.d/* |more
}


rgrep ()     # Bash shell syntax
{
  find $2 -type file -exec grep $1 {} \; | more
}
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Positional Parameters and Functions

```
$ killit oracle
Note: This will kill all the ‘oracle’ user processes.

$ rcgrep sed
/etc/init.d/pppd:     sed -e
's/^#.*//;s/\([^\\]\)#.*/\1/;s/[ ]*$//;s
/^[   ]*//' \
/etc/init.d/pppd:     sed -e
's/^#.*//;s/\([^\\]\)#.*/\1/;s/[ ]*$//;s
/^[   ]*//' \
/etc/init.d/README:scripts. The S* scripts should only be
used for cleanup during
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Positional Parameters and Functions

```
$ rgrep root /etc/default
# If CONSOLE is set, root can only login on that device.
# If the specified device is /dev/console, then root can
also log into
# Comment this line out to allow remote login by root.
# SUPATH sets the initial shell PATH variable for root
# to log all root logins at level LOG_NOTICE and multiple
failed login
# CONSOLE determines whether attempts to su to root should
be logged
# SUPATH sets the initial shell PATH variable for root
# root, LOG_INFO messages are generated for su's to other
users, and LOG_CRIT
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Return Values

- The `return` statement terminates the function and passes a value back to the calling shell or script.
- The `return` statement returns any designated value between `0` (zero) and `255`.
- By default, the value passed by the `return` statement is the current value of the `?` exit status variable.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Quiz

Positional parameters passed to functions are not the same positional parameters that are passed to a script.

a. True

b. False

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Agenda

- Creating user-defined functions in a shell script
- Using the `typeset` and `unset` statements in a function
- Autoloading a function file into a shell script

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The `typeset` and `unset` Statements

- The following shows the syntax for using `typeset` and `unset` with functions:
  - `typeset –f`: Lists the known functions and their definitions
    - `functions` is an alias for `typeset –f`
  - `typeset +f`: Lists the known function names
  - `unset -f` *`name`*: Unsets the value of the function
- The examples in the following slides use `typeset` and `unset` with functions.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The typeset and unset Statements

```
$ typeset -f
function killit
{
pkill -u $1
print -n "Had to kill process for user: $1 "
print "on $(date +%D) at $(date +%T)"
# The previous print statement may be appended to a log
file.
}
function rcgrep
{
grep $1 /etc/init.d/* |more
}
function rgrep
{
find $2 -type file -exec grep $1 {} \; | more
}
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The typeset and unset Statements

```
$ typeset +f
killit
rcgrep
Rgrep

$ alias | grep fun
functions='typeset -f'

$ unset -f rcgrep

$ typeset +f
killit
rgrep
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Agenda

- Creating user-defined functions in a shell script
- Using the `typeset` and `unset` statements in a function
- Autoloading a function file into a shell script

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Function File

- You can create functions within a shell script, or they can be external to the shell script, such as in a file.
- Only one function can be in each function file.
- A function file can be autoloaded into a shell script and used by that script.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Autoloading a Function File

- To autoload a function file:
  - Declare the FPATH variable before any command lines attempt to invoke a function from one of the function files

```
$ FPATH=$HOME/function_dir ; export FPATH
```

  - The directories listed in the FPATH environment variable should contain only function files
- By using the FPATH variable, the functions can be autoloaded into a shell script and do not need to be declared in every script.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Autoloading a Function File

```
$ cat holdertest.sh
#!/bin/bash
# Script name: holdertest.sh

FPATH=./funcs
export FPATH

print "Calling holder...“
holder

print
print "After the function var1 is: $var1“

$ ./holdertest.sh
Calling holder...
Type some text to continue: shell scripts
In function holder var1 is: shell scripts
After the function var1 is: shell scripts
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Create user-defined functions in a shell script
- Use the `typeset` and `unset` statements in a function
- Autoload a function file into a shell script

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 12 Overview: Functions

This practice covers the following topic:

- Using Functions

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.