1.109.2 Customize or write simple scripts Weight 3

Linux Professional Institute Certification — 102

Nick Urbanik <nicku@nicku.org>
This document Licensed under GPL—see section 18

2005 November

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

The snepang: #!

executable

script SUID?

rue and False

Shell Variable

Special Variables

uoting

Command Substitution

ne ii statement

hile statement

ne ior statement

ne test progran

Arithmetic

nput & Output

Alerting about problems by email

cense Of Tr ocument

Outline

Context Objectives The shebang: #! Making the script executable Should you make a script SUID? True and False Shell Variables Special Variables Quoting Quoting and Funny Chars Quoting

Command Substitution The if statement while statement The for statement The test program Conditions Arithmetic Input & Output Output with echo Input with read Alerting about problems by email License Of This Document

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Contex

Objectives

The Shebang.

Should you

rue and False

hell Variables

ecial Variables

oting

Command Substitution

he if statement

hile statement

ne for statement

rne test progran

Arithmeti

Input & Outpu

Alerting about problems by ema

icense Of T Document

Topic 109 Shells, Scripting, Programming and Compiling [8] Where we are up to

1.109.1 Customize and use the shell environment [5]

1.109.2 Customize or write simple scripts [3]

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

ne shebang: #!

executable

Should you make a script SUID?

rue and False

Silon variables

Special Variables

uoting

Command Substitution

he if statement

hile statement

ho for statement

e test program

The case program

Antimiette

nput & Output

Alerting about problems by email

ocument

Description of Objective

1.109.2 Customize or write simple scripts [3]

Candidate should be able to customize existing scripts, or write simple new (ba)sh scripts. This objective includes using standard sh syntax (*loops*, *tests*), using command substitution, testing command return values, testing of file status, and conditional *mailing to the superuser*. This objective also includes making sure the correct interpreter is called on the first (#!) line of scripts. This objective also includes managing *location*, *ownership*, execution and *suid-rights* of scripts.

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

Making the scrip executable

script SUID?

Shell Variables

Quoting

Command Sub

while statement

The for statemer

The test pro

Arithmet

Input & Outpu

Alerting about problems by email

cense Of 11 ocument

Key files, terms, and utilities include:

1.109.2 Customize or write simple scripts [3]

while — shell builtin: does things repetively while a condition is true

for — shell builtin: does things repetively, once with each element of a list

test — used to construct a condition

chmod — an external command, to change the permission on a file

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

ne shebang: #

Making the script executable

cript SUID?

ilue allu raise

Shell Variables

Quoting

Command Substitution

Command Substitution

hile statement

he for etetemor

he test program

Arithmetic

Innut ^o Outnu

Alerting about problems by ema

License Of TI Document

The Shebang: #!

- 1.109.2 Customize or write simple scripts Weight 3
 - Nick Urbanik
- Context
- Objective

The shebang: #!

Making the script executable

cript SUID?

ilue aliu raise

oting

Command Substitution

The if statement

hile statement

he test program

The test program

Arithmeti

Input & Outpu

Alerting about problems by ema

Document Document

- You ask the Linux kernel to execute the shell script
- kernel reads first two characters of the executable file
 - ▶ If first 2 chars are "#!" then
 - kernel executes the name that follows, with the file name of the script as a parameter
- ▶ Example: a file called find.sh has this as the first line:
 - #! /bin/sh
- then kernel executes this:

/bin/sh find.sh

- What will happen in each case if an executable file begins with:
 - #! /bin/rm
 - ▶ #! /bin/ls

Making the script executable

To easily execute a script, it should:

- ▶ be on the PATH
- have execute permission.

How to do each of these?

Red Hat Linux by default, includes the directory ~/bin on the PATH, so create this directory, and put your scripts there:

```
$ mkdir ~/bin ←
```

If your script is called script, then this command will make it executable:

```
\$ chmod +x script \hookleftarrow
```

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

Making the script executable

Snould you mak script SUID?

ue and False

Shell Variables

pecial Variables

ioting

Command Substitution

e if statement

niie Statement

e test nrogram

The test progra

Anumeuc

Input & Outpu

Alerting about problems by ema

icense Of T Document

Should you make a script SUID?

- Normally, when you run a script, the process is owned by you, and has the same access rights as you
- If a script has the SUID permission, then:
 - it does not matter who executes it!
 - the owner of the process is the owner of the file
 - This is very dangerous, especially if the owner of the file is root!
- Never make a shell script SUID, unless you really, really know what the risks are and how to avoid them
- Instead, write it in a language such as Perl, with taint checking, and make it as simple as possible.
- ► See Topic 1.114.1 Perform security administration tasks for details of manipulating SUID/SGID permissions.

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

Making the script executable

Should you make a script SUID?

rue and False

Shell Variables

.....

ammand Substitution

he if statement

while statement

The test progra

A with meaning

Input & Outp

Alerting about problems by ema

License Of Th Document

True and False

- Shell programs depend on executing external programs
- When any external program execution is successful, the exit status is zero, 0
- An error results in a non-zero error code
- ► To match this, in shell programming:
 - The value 0 is true
 - any non-zero value is false
- ▶ This is opposite from other programming languages

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

Asking the script

Should you r

True and False

Shell Variables

oting

Command Substitution

a if statement

nile statement

.

he test progra

Arithmetic

Input & Output

Alerting about problems by ema

license Of T Document

Variables—1

- Variables not declared; they just appear when assigned to
- Assignment:
 - no dollar sign
 - no space around equals sign
 - examples:
 - \$ x=10 # correct
 - \$ x = 10 # wrong: try to execute program called "x"
- ▶ Read value of variable:
 - put a '\$' in front of variable name
 - example:
 - \$ echo "The value of x is \$x"

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

The Shebang. #

executable

cript SUID?

rue and False

Shell Variables

Special Variables

oting

Command Substitution

hile statement

nite Statement

he test program

The cest progr

Innut 9 Outou

Alerting about

Variables—Assignments

- ➤ You can put *multiple assignments* on one line: i=0 j=10 k=100
- You can set a variable temporarily while executing a program:
 - \$ echo \$EDITOR
 emacsclient
 - \$ EDITOR=gedit crontab -e
 - \$ echo \$EDITOR
 emacsclient.

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

The shebang. #

Should you make

Shell Variables

Onon variables

....

Command Substitution

ne if statement

hile statement

ne ior statement

he test program

Arithmetic

Input & Outpu

Alerting about problems by ema

License Of 1 Document

Variables—Local to Script

- Variables disappear after a script finishes
- Variables created in a sub shell disappear
 - parent shell cannot read variables in a sub shell
 - example:

```
$ cat variables
#! /bin/sh
echo $HOME
HOME=happy
echo $HOME
$ ./variables
/home/nicku
happy
$ echo $HOME
```

/home/nicku

1.109.2 Customize or write simple scripts Weight 3

Nick Hrbanik

Objectives

ne snebang: #

executable

script SUID?

True and False

Shell Variables

-----9

Command Substitution

no n otatomont

while statement

he for statement

The test progra

Arithmeti

Input & Outpu

Alerting about problems by ema

License Of 1 Document

Variables—unsetting Them

You can make a variable hold the null string by assigning it to nothing, but it does not disappear totally: \$ VAR= ←

```
$ env | grep '^VAR' ← VAR=
```

You can make it disappear totally using unset:

```
$ unset VAR ←
```

\$ env | grep '^VAR' ←

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

Making the script

script SUID?

True and False

Shell Variables

Special variables

adung

Command Substitution

no ii statomont

nile statement

.

ne test prograr

Arithmetic

nput & Output

Alerting about problems by ema

license Of T Document

Command-line Parameters

- ► Command-line parameters are called \$0, \$1, \$2, ...
- Example: when call a shell script called "shell-script" like this:
 - \$ shell-script param1 param2 param3 param4 \hookleftarrow

variable	value
\$0	shell-script
\$1	param1
\$2	param2
\$3	param3
\$4	param4
\$#	number of parameters to the program, e.g., 4

Note: these variables are read-only.

1.109.2 Customize or write simple scripts Weight 3

Nick Hrhanik

Context

Objectives

he shebang: #

Should you m

ue and False

Special Variables

uotina

-- 3

ommand Substitution

e ii stateillellt

tre statement

e test progran

Arithmetic

Input & Output

Alerting about problems by ema

icense Of T locument

Special Built-in Variables

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objective

Making the script

script SUID?

True and False

.....

Special Variables

uoting

Command Substitution

The if statement

while statement

he for statement

The test progra

Arithmetic

Input & Outpu

Alerting about problems by email

Document Document

▶ Both \$@ and \$* are a list of all the parameters.

➤ The only difference between them is when they are quoted in quotes—see manual page for bash

\$? is exit status of last command

\$\$ is the process ID of the current shell

► Example shell script:

```
#! /bin/sh
echo $0 is the full name of this shell script
echo first parameter is $1
echo first parameter is $2
echo first parameter is $3
echo total number of parameters is $#
echo process ID is $$
```

Special Characters
Many characters have a special meaning to the shell

Character	Meaning
~	Home directory
`	Command substitution. Better: \$ ()
#	Comment
\$	Variable expression
&	Background Job
*	File name matching wildcard
	Pipe
(Start subshell
)	End subshell
[Start character set file name matching
]	End character set file name matching
{	Start command block
;	Command separator
\	Quote next character
,	Strong quote
"	Weak quote
<	Redirect Input
>	Redirect Output
/	Pathname directory separator
?	Single-character match in filenames
!	Pipline logical NOT
⟨space or tab⟩	shell normally splits at white space

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Quoting and Funny Chars

Quoting

- Sometimes you want to use a special character literally;
 i.e., without its special meaning.
- Called quoting
- ► Suppose you want to print the string: 2 * 3 > 5 is a valid inequality?
- ▶ If you did this:
 - \$ echo 2 * 3 > 5 is a valid inequality
 the new file '5' is created, containing the character '2',
 then the names of all the files in the current directory, then
 the string "3 is a valid inequality".

1.109.2 Customize or write simple scripts Weight 3

Nick Hrhanik

Context

Objectives

Making the script

hould you make a cript SUID?

frue and False

onell variables

pecial Variables

Quoting

Quoting and Funny Chars

Quoting

ioting

Sommand Substitution

ne ii statement

- ·

THE IOI Statemen

.

Alerting about

Quoting—2

- ➤ To make it work, you need to protect the special characters '*' and '>' from the shell by quoting them. There are three methods of quoting:
 - Using double quotes ("weak quotes")
 - Using single quotes ("strong quotes")
 - Using a backslash in front of each special character you want to quote
- This example shows all three:

```
$ echo "2 * 3 > 5 is a valid inequality"
```

- \$ echo '2 * 3 > 5 is a valid inequality'
- \$ echo 2 \times 3 > 5 is a valid inequality

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

Making the script executable

script SUID?

ue and raise

ieii variabies

Quoting Quoting and Funny Chars

Quoting

Sommand Substitution

e ii statement

while statement

he for statemer

The + - - + pregr

rithmetic

Inner d O Out

Alerting about problems by ema

Quoting—When to use it?

- Use quoting when you want to pass special characters to another program.
- Examples of programs that often use special characters:
 - ▶ find, locate, grep, expr, sed **and** echo
- Here are examples where quoting is required for the program to work properly:

```
$ find . -name \*.jpg
```

- \$ locate '/usr/bin/c*'
- \$ grep 'main.*(' *.c
- \$ i=\$(expr i * 5)

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

Making the script

Should you r script SUID?

ue and False

nieli variables

oting

Quoting and r

luoting

Johnnand Substitutio

ne if statement

-1-11 - atatamar

he for statemer

The test progra

rithmetic

Alerting about problems by em

More about Quoting

- Double quotes: "..." stop the special behaviour of all special characters, except for:
 - variable interpretation (\$)
 - backticks (`) see slide 5
 - ▶ the backslash (\)
- ▶ Single quotes ' . . . ' :
 - stop the special behaviour of all special characters
- Backslash:
 - preserves literal behaviour of character, except for newline; see slides §4, §9
 - Putting "\" at the end of the line lets you continue a long line on more than one physical line, but the shell will treat it as if it were all on one line.

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

Making the script

Should you m script SUID?

rue and False

Shell Variables

pecial Variables

Quoting and Funny Chars

Quoting

Command Substitution

e if statement

o ii statomont

he for statemen

rithmetic

a ta in io do

Alerting about

Command Substitution — \$ (...) or '...'

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

Should you ma

E I E . I . .

....

Quoting

oung

Command Substitution

The ii statement

while statement

he test program

.

Alerting about problems by em

Document

- ► Enclose command in \$ (...) or backticks: `... `
- ► Means, "Execute the command in the \$ (...) and put the output back here."
- ► Here is an example using expr:

```
$ expr 3 + 2
```

5

\$ i=expr 3 + 2 # error: try execute command '3'

\$ i=\$(expr 3 + 2) # correct

\$ **i='expr 3 + 2'** # also correct

Command Substitution—Example

We want to put the output of the command hostname into a variable:

```
$ hostname
```

nicku.org

\$ h=hostname

\$ echo \$h

hostname

- Oh dear, we only stored the name of the command, not the output of the command!
- Command substitution solves the problem:

```
$ h=$(hostname)
```

\$ echo \$h
nicku.org

▶ We put \$ (...) around the command. You can then assign the output of the command.

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objective

The shebang: #

Making the script executable

cript SUID?

ue and False

Shell Variables

eciai variables

Command Substitution

while statement

while statement

he test progran

Arithmetic

. . . .

Alerting about

icense Of TI Document

if Statement

Syntax:

```
if \langle test-commands \rangle then
   \langle statements-if-test-commands-1-true \rangle elif \langle test-commands-2 \rangle then
   \langle statements-if-test-commands-2-true \rangle else
   \langle statements-if-all-test-commands-false \rangle fi
```

Example:

```
if grep nick /etc/passwd > /dev/null 2>&1
then
    echo Nick has a local account here
else
    echo Nick has no local account here
fi
```

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Contex

Objective:

The shebang: #!

Making the script executable

Should you mak script SUID?

rue and False

Shell Variables

pecial Variables

oting

Command Substitution

The if statement

hile statement

The for statemen

he test progran

Arithmet

Input & Output

Alerting about problems by ema

License Of 1 Document

while Statement

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

```
Syntax:
```

```
while \langle test-commands \rangle
do
\langle (loop-body-statements)
done
```

► Example:

```
i=0
while [ "$i" -lt 10 ]
do
    echo -n "$i "  # -n suppresses newline.
    let "i = i + 1"  # i=$(expr $i + 1) also works
done
```

Context

Objectives

The shebang: #!

Making the script executable

Should you make

rue and False

hell Variables

Special Variables

uoting

Command Substitution

.....

while statement

The for statement

he test progran

Arithmetic

Input & Outpu

Alerting about problems by emai

license Of T Document

for Statement

Syntax:

```
for \langle name \rangle in \langle words \rangle
do
  \langle loop-body-statements \rangle
done
```

Example:

```
for planet in Mercury Venus Earth Mars \
    Jupiter Saturn Uranus Neptune Pluto
do
    echo $planet
done
```

► The backslash "\" quotes the newline. It's just a way of folding a long line in a shell script over two or more lines.

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

he shebang: #!

executable

script SUID?

rue and False

hell Variables

oting

ommand Substitution

he if statement

hile statement

The for statement

he test progran

Arithmetic

Input & Outpu

Alerting about problems by ema

license Of T Document

Making the script

Should you mak

True and False

Shell Variables

peciai variables

oung

Command Substitution

ie ii Statement

mile statement

The for statement

The test prograi

lancat 0 Oct

Alerting about problems by ema

Document Document

Here the shell turns *.txt into a list of file names ending in ".txt":

```
for i in *.txt
do
     echo $i
     grep 'lost treasure' $i
done
```

You can leave the in ⟨words⟩ out; in that case, ⟨name⟩ is set to each parameter in turn:

```
i=0
for parameter
do
    let 'i = i + 1'
    echo "parameter $i is $parameter"
done
```

statements and for while loops

Examples of string comparisons:

either test or [...]

"\$USER" = root 1

"\$USER" != root]

-z "\$USER"]

true if the string "\$USER" has zero length

string1 \< string2 | # true if string1 sorts less than string2 string1 \> string2] # true if string1 sorts greater than string2

Note that we need to quote the '>' and the '<' to avoid</p> interpreting them as file redirection.

All programming languages depend on conditions for if

true if the value of \$USER is "root"

true if the value of \$USER is not "root"

Shell programming uses a built-in command which is

- Note: the spaces after the "[" and before the "]" are essential.
- Also spaces are essential around operators

Conditions

Conditions—Integer Comparisons

Examples of *numeric* integer comparisons:

```
[ "xx" -eq 5 ] # true if the value of $x is 5 [ "xx" -ne 5 ] # true if integer $x is not 5 [ "xx" -lt 5 ] # true if integer $x is < 5 [ "xx" -gt 5 ] # true if integer $x is > 5 [ "xx" -le 5 ] # true if integer $x is x is x 5 [ "xx" -ge 5 ] # true if integer $x is x 5
```

- Note again that the spaces after the "[" and before the "]" are essential.
- Also spaces are essential around operators

1.109.2 Customize or write simple scripts Weight 3

Nick Hrhanik

Context

Objectives

ne snebang: # !

executable

script SUID?

iue anu i aise

Shell Variables

...

Command Substitution

o ii statomont

ile statement

ie ioi statement

The test program

Arithmetic

Alerting about

icense Of Th ocument

Conditions—File Tests, NOT Operator

- The shell provides many tests of information about files.
- ▶ Do man test to see the complete list.
- Some examples:

```
[ -f file ]
              # true if file is an ordinary file
              # true if file is NOT an ordinary file
  ! -f file l
 -d file 1
              # true if file is a directory
[ -u file ]
              # true if file has SUID permission
 -a file 1
              # true if file has SGID permission
[ -x file ]
              # true if file exists and is executable
[ -r file ]
              # true if file exists and is readable
 -w file 1
              # true if file exists and is writeable
[ file1 -nt file2 ] # true if file1 is newer than file2
```

- Note again: the spaces after the "[" and before the "]" are essential.
- Also spaces are essential around operators

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objective:

ne snebang: # !

Making the scripexecutable

cript SUID?

ue and False

Shell Variables

· Juotina

Command Substitutio

The :f -t-t----

while statement

.....

he test progran

Conditions Arithmetic

Arithmetic

Input & Out

Alerting about problems by email

cense Of Th ocument

Conditions—Combining Comparisons

Examples of combining comparisons with AND: -a and OR: -o, and grouping with \ (...\)

```
# true if the value of \$x is 5 AND \$USER is not equal to root: [ "\$x" -eq 5 -a "\$USER" != root ] # true if the value of \$x is 5 OR \$USER is not equal to root: [ "\$x" -eq 5 -o "\$USER" != root ] # true if (the value of \$x is 5 OR \$USER is not equal to root ) AND # (\$y > 7 OR \$HOME has the value happy) [ \ ("\$x" -eq 5 -o "\$USER" != root \) -a \ \ ("\$y" -gt 7 -o "\$HOME" = happy \) ]
```

- Note again that the spaces after the "[" and before the "]" are essential.
- Do man test to see the information about all the operators.

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

laking the scrip

cript SUID?

True and Fals

Shell Variables

Special Variables

. . . .

Command Substitution

hile statement

niie statement

he test program

Conditions

Arithmetic

Input & Ou

Alerting about

icense Of Th locument

Arithmetic Assignments

- Can do with the external program expr
 - ... but expr is not so easy to use, although it is very standard and portable: see man expr
 - Easier is to use the built in let command
 - see help let
 - Examples:

```
$ let x=1+4
$ let ++x  # Now x is 6
$ let x='1 + 4'
$ let 'x = 1 + 4'
$ let x="(2 + 3) * 5"  # now x is 25
$ let "x = 2 + 3 * 5"  # now x is 17
$ let "x += 5"  # now x is 22
$ let "x = x + 5"  # now x is 27; NOTE NO
```

- Notice that you do not need to quote the special characters with let.
- Quote if you want to use white space.
- Do not put a dollar in front of variable, even on right side of assignment; see last example.

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objective

The Shebang. #

executable

Shell Variables

Special Variables

uoting

The if statement

while statement

THE TOT STATETHETT

The test program

Arithmetic

Input & Outpu

Alerting about problems by ema

icense Of T locument

Output with echo

- ► To perform output, use echo, or for more formatting, printf.
- ▶ Use echo -n to print no newline at end.
- ▶ Just echo by itself prints a newline

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objective

The shebang: #!

Making the script executable

script SUID?

irue and Faise

Shell variables

Special Variables

uoting

Command Substitution

ne ir statement

hile statement

mile statement

e test progran

ne test progran

Arithmetic

Input & Output
Output with echo

Input with read

roblems by em

Input: the read Command

- For input, use the built-in shell command read
- read reads standard input and puts the result into one or more variables
- ▶ If use one variable, variable holds the whole line
- Syntax:

```
read \langle var1 \rangle \dots
```

▶ Often used with a while loop like this:

```
while read var1 var2
do
    # do something with $var1 and $var2
```

done

Loop terminates when reach end of file

To prompt and read a value from a user, you could do:

```
while [ -z "$value" ]; do
    echo -n "Enter a value: "
    read value
done
# Now do something with $value
```

1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

biectives

The shebang: #!

laking the script kecutable

ript SUID?

rue and False

Sanaial Variables

oting

Command Substitution

hile statement

hile statement

ne test program

. ...

Arithmetic

Input & Output
Output with echo
Input with read

roblems by ema

1.109.2

Customize or write

```
| grep -i "shell script" | awk -F: '{print $1}'

Let's see how this works. I suggest executing the commands

The shell
```

grep -i "shell script"

```
separately to see what they do:
```

\$ file /bin/* /usr/bin/*
\$ file /bin/* /usr/bin/*

```
$ file /bin/* /usr/bin/* | grep -i "shell script" | awk -F:

The awk program is actually a complete programming
```

text.

awk automatically loops through the input, and divides the input lines into fields. It calls these fields \$1.52. SNE.50

language. It is mainly useful for selecting columns of data from

input lines into fields. It calls these fields \$1, \$2,...\$NF. \$0 contains the whole line. Here the option -F: sets the *field separator* to the colon character. Normally it is any white space. So printing \$1 here prints what comes before the colon, which is the file name.

Suppose you want to look for all shell scripts containing a particular command or statement? Looking for example shell scripts that use the mktemp command:

t

shebang: #!

king the script
cutable

'{printals\$1}'

uoting
ommand Substitution

ile statement

ithmetic

nput & Output
Output with echo
Input with read

erting about oblems by emai

```
$ file /bin/* /usr/bin/* /usr/sbin/* /sbin/* /etc/rc.d/* /usr/X11R6/bin/*
 grep -i 'shell script' | awk -F: '{print $1}' | xargs grep mkt@simple scripts
                                                                         Weight 3
                                                                        Nick Urbanik
```

Alerting about problems by email

Alerting about problems by email

#! /bin/sh

```
1.109.2
Customize or write
  simple scripts
    Weight 3
  Nick Hrhanik
```

Alerting about problems by email

```
# A quick script whipped up by Nick to send mail if
# root file system is more than 90 per cent full.

percentful=$(df / | awk 'NR > 1{sub("%", "", $5);print $5}')
if [ "$percentful" -gt 90 ]
then
    message="root file system is $percentful% full"
    echo "$message" | mail -s $message root
fi
```

Topics Covered

Context Objectives The shebang: #! Making the script executable Should you make a script SUID? True and False Shell Variables Special Variables Quoting Quoting and Funny Chars Quoting

Command Substitution The if statement while statement The for statement The test program Conditions Arithmetic Input & Output Output with echo Input with read Alerting about problems by email License Of This Document 1.109.2 Customize or write simple scripts Weight 3

Nick Urbanik

Context

Objectives

The shebang: #

Making the script executable

script SUID?

ide and raise

nocial Variables

oting

Command Substitution

he if statement

hile statement

o for statemen

he test program

Arithmetic

Input & Output

Alerting about problems by email

icense Of TI Document

License Of This Document

Copyright © 2005 Nick Urbanik <nicku@nicku.org>
You can redistribute modified or unmodified copies of this
document provided that this copyright notice and this
permission notice are preserved on all copies under the terms
of the GNU General Public License as published by the Free
Software Foundation — either version 2 of the License or (at
your option) any later version.

1.109.2 Customize or write simple scripts

Weight 3
Nick Urbanik

Context

Objectives

Making the script

Should you m

True and False

Shell Variables

Special Variable

uoting

Command Substitution

he if statement

ile statement

rie statement

e test progran

.

Arithmetic

nnut & Output

Alerting about problems by emai

License Of This Document