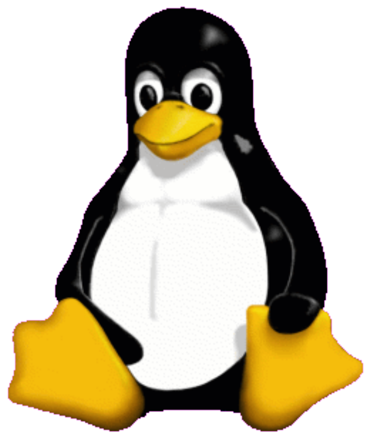
Unix104

BIOL647
Digital Biology

Rodolfo Aramayo



Unix104

Fundamentals of Scripting

Understanding Bash script syntax

```
## Check $GENOME
if [[ ! -d $GENOME ]]; then
        echo "Could not find reference genome: $GENOME, exiting. Please make sure
        exit 1
else
        GENOME_GTF="$GENOME/Annotation/Genes/genes.gtf"
        GENOME_FA="$GENOME/Sequence/WholeGenomeFasta/genome.fa"
        if [[ ! -f $GENOME_GTF ]]; then
                echo "$GENOME_GTF not found, exiting"
                exit 1
        fi
        if [[ ! -f $GENOME_FA ]]; then
                echo "$GENOME_FA not found, exiting"
                exit 1
        fi
        STAR_INDEX="$GENOME/star/STAR_2.4.1c/"
fi
```

- ***Variables***
- ***Arguments***
- ***Flow-control logic***

Unix104

Fundamentals of Scripting
Building Bash Scripts

To Shebang or not to Shebang

- ***What is The Shebang?***
    - ***In computing, a shebang is the character sequence consisting of the characters number sign and exclamation mark (#!) at the beginning of a script. It is also called sha-bang***

```
# Hashbang or Shebang, tells the shell that this is a BASH script \
# and that it should be run as such



# BASH
#!              <—Hash—bang
/usr/bin/bash   <—PATH to Bash executable

#!/usr/bin/bash
...
...
...
...
Exit 0
```

Unix104

Fundamentals of Scripting
Building Bash Scripts

To Shebang or not to Shebang

```
# Hashbang or Shebang, tells the shell that this is a PYTHON script \
# and that it should be run as such


# PYTHON
#!                  <—Hash—bang
/usr/bin/python3    <—PATH to Python3 executable

#!/usr/bin/python3
...
...
...
...
```

Unix104

Fundamentals of Scripting
Building Bash Scripts

To Shebang or not to Shebang

```
# Hashbang or Shebang, tells the shell that this is a PERL script \
# and that it should be run as such


# PERL
#!               <-Hash-bang
/usr/bin/perl   <-PATH to PERL executable

#!/usr/bin/perl
...
...
...
...
```

Unix104

Fundamentals of Scripting

Introducing for loops

- The for loop is used to execute commands a finite number of times on a list of items.
- The for command is followed by a user-defined variable, the keyword in, and then a list of words.
- The variable assumes the value of an element from the list of words, in order from left to right, before each iteration of the loop.
- At the end of each iteration, the current value gets shifted off from the beginning of the list and the next word becomes the head of the list.
- The loop finishes when there are no more words left in the word list.

Unix104

Fundamentals of Scripting

Introducing for loops

```
# Loop 01
for i in 1 2 3;do
 echo $i;
done


# Loop 02
for i in {1..10};do
 echo $i;
done


# Loop 03
for i in {1..100..2};do
 echo $i;
Done
```

Unix104

Fundamentals of Scripting

Introducing for loops

- ***01script.sh***

```
for i in {1..50};do
 echo "echo "$i"";
done
# Note how we are escaping special characters

for i in {1..50};do
 echo "echo \"$i\"";
done

# Execute file as follows:
./01script.sh and bash ./01script.sh

# Now add as a header to the file:
#!/path/to/bash

# And make file 01script.sh executable:
chmod 744 01script.sh

# And execute file as follows:
./01script.sh and bash ./01script.sh
```

Unix104

Fundamentals of Scripting

Working with variables

- ***02script.sh***

```
#!/usr/bin/bash
# 02script.sh

a=hello              # Note No Spaces
b="Good Morning"  # Note the quotes
c=42
# To use these variables we call them with the '$' sign
echo $a
echo $b
echo $c

exit 0
```

Unix104

Fundamentals of Scripting

Working with variables

- ***03script.sh***

```
#!/usr/bin/bash
# 03script.sh

a=hello            # Note No Spaces
b="Good Morning"   # Note the quotes
c=44
# To use these variables we call them with the '$' sign
echo $a
echo $b
echo $c

echo "$b! I have $c apples" <—NOTE The presence of spaces

exit 0
```

Unix104

Fundamentals of Scripting

Adding attributes to variables

- ***04script.sh***

```
#!/usr/bin/bash
# 04script.sh

declare -i d=var1     # -I = The variable is to be treated as an integer
                      # d must be an integer

echo "d=$d"           # Prints '0' because it is empty

declare -i d=12
echo "d=$d"           # Prints '12' because variable 'd' was declared to be '12'

d=hello               # Variable redefined as a 'non-integer'
echo "d=$d"           # Prints '0' because variable 'd' can only be an integer

echo "------------"   # Print a spacer

declare -r e=var2     # -r Make names readonly
                      # e must be read-only

echo "e=$e"           # Prints 'var2' because it is read-only

declare -r e
e=13
echo "e=$e"           # Fails to assign a new value to the variable 'e' because it is read-only

exit 0
```

Unix104

Fundamentals of Scripting

Built-in variables

- ***05script.sh***

```
#!/usr/bin/bash
# 05script.sh

# Returns Home directory
echo $HOME

# Returns current directory
echo $PWD

# Returns machine type
echo $MACHTYPE

# Returns system name
echo $HOSTNAME

# Returns Bash version
echo $BASH_VERSION

# Returns the number of seconds the Bash session has run
# Inside a script it counts the seconds since the script started (timing)
echo $SECONDS

# Returns the name of the script
echo $0

exit 0
```

Unix104

Fundamentals of Scripting

Command substitution

- ***06script.sh***

```
#!/usr/bin/bash
# 06script.sh

a=$HOME
echo $a

b=$(pwd)
echo $b

exit 0
```

Unix104

Fundamentals of Scripting

Command substitution

- *07script.sh*

```
#!/usr/bin/bash
# 07script.sh

a=$(ping -c 1 google.com | grep 'bytes from' | cut -d = -f 4)

echo "The ping was $a"

exit 0
```

Unix104

Fundamentals of Scripting

Arithmetic operations

```
      (( expression ))
val=$(( expression )) # Note the '$' sign
|----------------|----------|
| Operation      | Operator |
|----------------|--------- |
| Exponentiation | $a ** $b |
| Multiplication | $a * $b  |
| Division       | $a / $b  |
| Modulo         | $a % $b  |
| Addition       | $a + $b  |
| Subtraction    | $a - $b  |
|----------------|----------|
```

Unix104

Fundamentals of Scripting

Arithmetic operations

- ***08script.sh***

```
#!/usr/bin/bash
# 08script.sh

d=2

e=$((d+2))

echo "The value of e is $e"

exit 0
```

Unix104

Fundamentals of Scripting

Arithmetic operations

- ***09script.sh***

```
#!/usr/bin/bash
# 09script.sh

d=2
e=$((d+2))
echo "The value of e is $e"
((e++))
echo "The value of e is $e"
((e--))
echo "The value of e is $e"
((e+=5))
echo "The value of e is $e"
((e*=3))
echo "The value of e is $e"
((e/=3))
echo "The value of e is $e"
((e-=5))
echo "The value of e is $e"

exit 0
```

Unix104

Fundamentals of Scripting

Arithmetic operations

- ***10script.sh***

```
#!/usr/bin/bash
# 10script.sh

f=(1/3)

echo "The value of e is $f"

exit 0
```

Unix104

Fundamentals of Scripting

Arithmetic operations

- ***11script.sh***

```
#!/usr/bin/bash
# 11script.sh

f="1/3"
echo "The value of e is $f"

g=$(echo "1/3" | bc -l)
echo "The value of g is $g"

exit 0
```

Unix104

Fundamentals of Scripting

Comparing values

- ***Why 0 is true but false is 1 in the shell?***

```
[[ expression ]]
     0 = True
    1 = False
```

Unix104

Fundamentals of Scripting

Logical Arithmetic operators

Operation	Operator
Less than	`[[ $a -lt $b ]]`
Greater than	`[[ $a -gt $b ]]`
Less than or equal to	`[[ $a -le $b ]]`
Greater than or equal to	`[[ $a -ge $b ]]`
Equal	`[[ $a -eq $b ]]`
Not equal	`[[ $a -ne $b ]]`

Unix104

Fundamentals of Scripting

Logical Comparison operators

- ***12script.sh***

```
#!/usr/bin/bash
# 12script.sh

[[ "20" -gt "100" ]]
echo $?

[[ "20" -lt "100" ]]
echo $?

[[ "200" -gt "100" ]]
echo $?

[[ "200" -lt "100" ]]
echo $?

[[ "200" -eq "200" ]]
echo $?

exit 0
```

Unix104

Fundamentals of Scripting

Logical Comparison operators

Operation	Operator
Logical AND	`[[ $a && $b ]]`
Logical OR	`[[ $a || $b ]]`
Logical NOT	`[[ ! $a ]]`
Is Null?	`[[ -z $a ]]`
Is Not Null?	`[[ -n $A ]]`

Unix104

Fundamentals of Scripting

Logical Comparison operators

- ***13script.sh***

```
#!/usr/bin/bash
# 13script.sh

a=""

b="cat"

[[ -z $a && -n $b ]]

echo $?

exit 0
```

Unix104

Fundamentals of Scripting

Working with strings

- ***Using the command-line:***

```
> a="Hello" && b="World" && c=$a$b

> echo $c

> echo ${c}
# To Find out how long the string is:
> echo ${#c}
# To request a substring:
> d=${c:3}

> echo $d

# To request a specific number of characters after that position asking at character 3 and asking for 4 characters after that
> e=${c:3:4}
> echo $e
> echo ${c}
> echo ${#c}
# To request the last 10 letters
> echo ${c: -10}
# To request the last 8 letters
> echo ${c: -8}

# To request the last 2 letters of the last 8 letters
> echo ${c: -8:2}
```

Unix104

Fundamentals of Scripting

Working with strings

- ***To replace text in a string with some other text:***

```
fruit="apple banana banana cherry"

# To replace text in a string with some other text in this case banana with durian or the first instance of the search
> echo ${fruit/banana/durian}

# To replace all instances of banana with durian
> echo ${fruit//banana/durian}

# To replace the term only of the term is the very beginning of the string
> echo ${fruit/#apple/durian}

# and it only works if the term is at the beginning of the string
> echo ${fruit/#banana/durian}

# To replace the term only of the term is the very end of the string
> echo ${fruit/%cherry/durian}

# and it only works if the term is at the end of the string
> echo ${fruit/%banana/durian}

# Using matching terms
> echo ${fruit/c*/durian}
```

Unix104

Fundamentals of Scripting

Working with strings

• ***Using basename and dirname:***

```
> "$(basename $File .fq)"

> a="/root/dir01/dir02/SRA12345.fastq"
> echo $a
> echo $a $(basename $a)
> echo $a $(dirname $a)

# To delete fastq
> echo $a $(basename $a fastq)

# To add a new name
> echo $a $(basename $a fastq)fq
```

Unix104

Fundamentals of Scripting

Working with strings

- ***Using Parameter Expansion:***

```
> param="racecar"

# Extraction: offset = 3, length = 2
# ${param:offset:length}

> echo ${param}

> echo ${param:3}

> echo ${param:3:2}
```

Unix104

Fundamentals of Scripting

Working with strings

- ***Using Parameter Expansion:***

```
> param="racecar"

# Removal from left edge: pattern = “*c”
# ${param#pattern}

> echo ${param#*c}

> echo ${param##*c}

# Removal from right edge: pattern = “c*”
# ${param%pattern}

> echo ${param%c*}

> echo ${param%%c*}
```

Unix104

Fundamentals of Scripting

Coloring and styling text

- ***Colored text (using ANSI codes)***

Color	Foreground	Background
Black	30	40
Red	31	41
Green	32	42
Yellow	33	43
Blue	34	44
Magenta	35	45
Cyan	36	46
White	37	47

Unix104

Fundamentals of Scripting

Coloring and styling text

- ***White on Black***

```
# -e                <-Allows escaping characters
# \033[37;40m       <-Escaped sequence
# Color Text        <-String to print out
# \033[0m           <-Reset the colors


> echo -e '\033[37;40mColor Text\033[0m'
```

Unix104

Fundamentals of Scripting

Coloring and styling text

• ***Black on Red:***

```
# -e                <-Allows escaping characters
# \033[30;41m       <-Escaped sequence
# Color Text        <-String to print out
# \033[0m           <-Reset the colors



> echo -e '\033[30;41mColor Text\033[0m'
```

Unix104

Fundamentals of Scripting

Coloring and styling text

- ***Green on Black:***

```
# -e                <-Allows escaping characters
# \033[32;40m       <-Escaped sequence
# Color Text        <-String to print out
# \033[0m           <-Reset the colors


> echo -e '\033[32;40mColor Text\033[0m'
```

Unix104

Fundamentals of Scripting

Coloring and styling text

- ***Red on White:***

```
# -e                <-Allows escaping characters
# \033[31;47m       <-Escaped sequence
# Color Text        <-String to print out
# \033[0m           <-Reset the colors


> echo -e '\033[31;47mColor Text\033[0m'
```

Unix104

Fundamentals of Scripting

Coloring and styling text

- ***Blue on Yellow:***

```
# -e                <-Allows escaping characters
# \033[34;43m       <-Escaped sequence
# Color Text        <-String to print out
# \033[0m           <-Reset the colors


> echo -e '\033[34;43mColor Text\033[0m'
```

Unix104

Fundamentals of Scripting

Coloring and styling text

- ***Styled text (ANSI)***

Style	Value
No Style	0
Bold	1
Low Intensity	2
Underline	4
Blinking	5
Reverse	7
Invisible	8

Unix104

Fundamentals of Scripting

Coloring and styling text

- ***Error Message:***

```
> echo -e '\033[5;31;42mERROR: \033[0m'

> echo -e '\033[5;31;42mERROR: \033[0m\033[31;40mSomething went wrong\033[0m'
```

Unix104

Fundamentals of Scripting

Working with arrays

• ***Simple arrays (Using the command-line):***

```
# Empty array
> a=()

# Array with 3 elements
> b=("apple" "banana" "cherry")

> echo ${b[2]}
> b[5]="kiwi"
> b+=("mango")

# To print all elements on the array
echo ${b[@]}

# To request the first element of the array
echo ${b[0]}

# To request the last element of the array
echo ${b[@]: -1}
```

Unix104

Fundamentals of Scripting

Working with arrays

- ***Associative arrays (Using the command-line):***

```
> declare -A myarray

> myarray[color]=Blue

> myarray["office" "building"]="TAMU Biology"

> echo ${myarray["office" "building"]} is ${myarray[color]}
```

Unix104

Fundamentals of Scripting

Reading and writing text files

- ***14script.sh***

```
# Using the command line:
> echo -e "Command01\nCommand02\nCommand03\nCommand04\nCommand05" > 14script.txt

# Script
#!/usr/bin/bash
# 14script.sh

while read i; do
   echo "$i";
done < 14script.txt

exit 0
```

Unix104

Fundamentals of Scripting

Using *here* documents

- ***15script.sh***

```
# Used a lot for entering documentation
# Avoids multiple ‘echo’ commands

#!/usr/bin/bash
# 15script.sh
# EndofText Must be unique

cat <<EOF
Line01
Line02
Line03
EOF

cat <<EOF
Text01
Text02
Text03
EOF

exit 0
```

Unix104

Fundamentals of Scripting

Control Structures

- ***Testing truth conditions with the if keyword***

```
'if' statatement
if [[ expression ]] or if (( expression )) or if statement
if [[ expression ]]; then echo "True"; else echo "False"; fi
```

Unix104

Fundamentals of Scripting

Control Structures

- ***Testing truth conditions with the if keyword***

```
> if [[ 2 -gt 3 ]]; then echo "True"; else echo "False"; fi


> if [[ 2 -gt 1 ]]; then echo "True"; else echo "False"; fi
```

Unix104

Fundamentals of Scripting

Control Structures

- ***Testing files existence***

```
> if [[ ! -f test ]]; then echo "The file test does not exist";fi

> if [[ ! -f test ]]; then echo "The file test does not exist";else echo "The file test does exist";fi

> touch test

> if [[ ! -f test ]]; then echo "The file test does not exist";else echo "The file test does exist";fi
```

Unix104

Fundamentals of Scripting

Control Structures

- ***Testing files content***

```
> if [[ ! -s test ]]; then echo "The file test is empty";else echo "The file test is not empty";fi

> echo "bla bla" > test

> if [[ ! -s test ]]; then echo "The file test is empty";else echo "The file test is not empty";fi
```

Unix104

Fundamentals of Scripting

Working with while and until loops

- ***16script.sh***

```
#!/usr/bin/bash
# 16script.sh

i=0

while [[ $i -le 10 ]];do
   echo i:$i
   ((i+=1));
done

exit 0
```

Unix104

Fundamentals of Scripting

Working with while and until loops

- ***17script.sh***

```
#!/usr/bin/bash
# 17script.sh

i=0

while [[ $i -le 10 ]];do
   echo i:$i
   ((i+=1));
done

j=0

until [[ $j -ge 10 ]];do
   echo j:$j
   ((j+=1));
done

exit 0
```

Unix104

Fundamentals of Scripting

Advanced for loops

- ***"C" Style***

```
# Simple Loops:

for (( i=1; i<=10; i++));do

   echo $i;

done

# With Arrays Loops:

arr=("Line01" "Line02" "Line03")

> for i in ${arr[@]};do echo $i;done

> for i in ${arr[0]};do echo $i;done

> for i in ${arr[2]};do echo $i;done
```

Unix104

Fundamentals of Scripting

Selecting behavior using case

- ***18script.sh***

```
#!/usr/bin/bash
# 18script.sh

a="dog"

case $a in
   cat)       echo "Feline";;
   dog|puppy) echo "Canine";;
   *)         echo "No Match";;
esac

exit 0
```

Unix104

Fundamentals of Scripting

Using Functions
To avoid repeating code blocks use functions

- ***19script.sh***

```
#!/usr/bin/bash
# 19script.sh

function test {
   echo "Robert"
   }

echo "And now I am greeting!"

test

exit 0
```

Unix104

Fundamentals of Scripting

Interacting with the user
Arguments

- ***20script.sh***

```
#!/usr/bin/bash
# 20script.sh

echo $0 # Name of the script
echo $1
echo $2
echo $@ # All variables
echo $# # Number of variables

exit 0
```

- ***Save 20script.sh and Run***
    - ***bash ./20script.sh***
    - ***bash ./20script.sh file01 file02 file03***

Unix104

Fundamentals of Scripting

Interacting with the user
Flags

- ***21script.sh***

```
#!/usr/bin/bash
# 21script.sh
args=("$@");
FILENAME00=${args[0]}
FILENAME01=${args[1]}
FILENAME02=${args[2]}

echo $FILENAME00
echo $FILENAME01
echo $FILENAME02

exit 0
```

- ***Save 21script.sh and Run***
  - ***bash ./21script.sh file01 file02 file03***
  - ***bash ./21script.sh file03 file02 file01***
  - ***bash ./21script.sh file03 file03 file03***

Unix104

Fundamentals of Scripting

Interacting with the user
Flags

- ***22script.sh***

```
#!/usr/bin/bash
# 22script.sh

while getopts u:p: option;do
   case $option in
      u) user=$OPTARG;;
      p) pass=$OPTARG;;
   esac
done
echo "User: $user / Passwd: $pass"
exit 0
```

- ***Save 22script.sh and Run***
    - ***bash ./22script.sh***
    - ***bash ./22script.sh -p secret -u rod***

Unix104

Fundamentals of Scripting

Interacting with the user
Flags

- ***23script.sh***

```
#!/usr/bin/bash
# 23script.sh
while getopts u:p:a:b: option;do
   case $option in
      u) user=$OPTARG;;
      p) pass=$OPTARG;;
      a) echo "Got the A Flag";;
      b) echo "Got the B Flag";;
   esac
done
echo "User: $user / Passwd: $pass"
exit 0
```

- ***Save 23script.sh and Run***
    - ***bash ./23script.sh -u rod -p secret***
    - ***bash ./23script.sh -p secret -u rod -a test***
    - ***bash ./23script.sh -p secret -u rod -b test***

Unix104

Fundamentals of Scripting

Interacting with the user
Flags

- ***24script.sh***

```
#!/usr/bin/bash
# 24script.sh
while getopts u:p:ab: option;do
   case $option in
      u) user=$OPTARG;;
      p) pass=$OPTARG;;
      a) echo "Got the A Flag";;
      b) echo "Got the B Flag";;
      ?) echo "I do not know what $OPTARG is!";;
   esac
done
echo "User: $user / Passwd: $pass"
exit 0
```

- ***Save 24script.sh and Run***
  - ***bash ./24script.sh -u rod -p secret***
  - ***bash ./24script.sh -p secret -u rod -a test***
  - ***bash ./24script.sh -p secret -u rod -b test***
  - ***bash ./24script.sh -p secret -u rod -a test -b test***

Unix104

Fundamentals of Scripting
Building Bash Scripts

- ***25script.sh***

```
#!/usr/bin/bash
# 25script.sh

function numberthings {
   i=1

   for f in $@;do
      echo $i: $f
      ((i+=1));
   done
}

numberthings $(ls)

exit 0
```

Unix104

BIOL647
Digital Biology

Rodolfo Aramayo