لغة Bash الدليل الشامل الأول

باش (Bourne Again SHell) هو لغة تحكم في نظام التشغيل Unix ومشتق من اللغة الشل (Shell). يوفر باش واجهة سطر الأوامر القوية (CLI) للتفاعل مع النظام وتوجيه المهام. إليك ملخصٌ للمفاهيم الأساسية في برمجة باش.

السيرفرات ولينكس أوامر لينكس

اسم الدليل: لغة Bash

رقم الدليل: 100

المجال: لينكس والسيرفرات

المتطلبات: أساسيات أوامر لينكس

مسار تعلم متعلق بهذا الدليل: مسار سكربتات Bash

الفيديو المتعلق بهذا الدليل: فيديو

تاريخ تحديث الدليل: مارس 26, 2024

فهرس الدليل

- نتيجة التنفيذ Exit Status
  - Exit 0
  - Exit failure
- الجمل الشرطية Conditional Statement
  - if statement
  - if-else statement
  - elif statement
  - معاملات المقارنة Comparison Operators
  - المعاملات المنطقية Logical Operator
- التكرار Loops
- الدوال Functions
  - المعاملات Parameters
  - المعاملات المُمررة Arguments
- المعاملات في لغة Bash
  - المعاملات الموضعية | Positional Parameters
  - المعاملات الخاصة | Special Parameters

لغة باش (Bourne Again SHell) هو لغة تحكم في نظام التشغيل Unix ومشتق من اللغة الشل (Shell). يوفر باش واجهة سطر الأوامر القوية (CLI) للتفاعل مع النظام وتوجيه المهام. إليك ملخّص للمفاهيم الأساسية في برمجة باش.

لغة Bash: أساسيات الأساسيات

فيما يلي الأساسيات المهمة التي يجب أن تعرفها في بداية تعلمك للغة باش.

الأوامر الأساسية

الأوامر هي التعليمات التي يتم إرسالها إلى Shell لأداء مهام محددة. فيما يلي أمثلة عليها:

```bash
# سرد الملفات في المسار الحالي
ls

# إنشاء مجلد جديد
mkdir newFolder

# إنشاء ملف جديد
touch newFile

# حذف ملف
rm file
```

Bash

اسم الأمر	توضيح
ls	يُستخدم لعرض قائمة بالملفات والمجلدات في الدليل الحالي.
mkdir	يُستخدم لإنشاء مجلد جديد.
rm	يُستخدم لحذف ملف أو مجلد.
cp	يُستخدم لنسخ ملف أو مجلد من موقع إلى آخر.
mv	يُستخدم لنقل ملف أو مجلد من موقع إلى آخر، أو لإعادة تسمية ملف أو مجلد
touch	يُستخدم لإنشاء ملف فارغ جديد أو لتحديث وقت تعديل ملف موجود.
cd	يُستخدم للتنقل بين المجلدات، أو للانتقال إلى مجلد معين.
grep	يُستخدم للبحث عن نص محدد داخل ملفات معينة أو نتائج الأوامر الأخرى.
cat	يُستخدم لعرض محتوى الملفات على الشاشة، أو لدمج ملفات مع بعضها البعض.
chmod	يُستخدم لتغيير أوامر التحكم في الوصول إلى الملفات والمجلدات (صلاحيات الملف).

أهم أوامر لينكس التي يجب عليك أن تتعلمها وتتقنها للاستمرار بتعلم لغة باش.

سكربتات Bash

النصوص البرمجية في باش هي ملفات تحتوي على سلسلة من الأوامر والتعليمات. يتم تنفيذ هذه النصوص من قبل مترجم باش. عادةً ما تكون النصوص البرمجية لها امتداد `.sh`، على الرغم من أن هذا ليس ضروريًا تمامًا.

> لاحظ أن يمكنك استخدام نفس أوامر لينكس المذكورة في الجدول السابق في سكربتات لغة Bash

إنشاء أول سكربت باش

1. **اختر محرر النصوص المناسب لك**: ابدأ بفتح محرر نصوص في نظام لينكس. يمكنك استخدام أي محرر نصوص تفضله، مثل Nano أو Vim أو Emacs.
2. **كتابة السكربت**: أول خطوة في كتابة السكربت هي إنشاء ملف جديد وإعطائه امتداد ".sh" ليدل على أنه سكربت باش. مثلاً، يمكنك تسمية ملفك "first-script.sh".
3. **كتابة الأوامر داخل السكربت**.
4. **جعل السكربت قابل للتنفيذ**: باستخدام أمر chmod
5. **تشغيل السكربت**

```bash
# Step 1, 2
# Terminal

nano first-script.sh
```

Bash

```bash
#!/bin/bash

# Step  3

# هذا سكربت باش بسيط
echo "مرحباً، هذا أول سكربت باش لي!"
```

Bash

```bash
# Step 4
# Terminal

chmod +x first-script.sh
```

Bash

```
```

```bash
# Step 5
# Terminal

./first-script.sh

مرحباً، هذا أول سكربت باش لي!

```

Bash

#!/bin/bash

السطر "#!/bin/bash" أو ما يُعرف بـ shebang هو عبارة توجد في أعلى ملفات السكربتات في نظام لينكس وبعض أنظمة التشغيل الأخرى. وتستخدم لتحديد مكان البرنامج المستخدم لتشغيل السكربت.

- الرمز "#!" يشير إلى أن السطر الذي يليه هو خاص بتعليمات التشغيل.
- "/bin/bash" يشير إلى موقع برنامج باش في النظام. في هذه الحالة، يُفترض أن السكربت سيتم تنفيذه باستخدام باش (Bash) كمترجم.

هذا السطر مهم لأنه يحدد البرنامج الذي يجب استخدامه لتنفيذ السكربت. فإذا كنت ترغب في تشغيل سكربت بلغة محددة مثل الباش، فيجب أن يكون هذا السطر موجودًا في البداية.

إذا كان السكربت يتوقع أن يتم تنفيذه باستخدام باش، فإن وجود هذا السطر مهم جدًا. فإذا كان السطر مفقودًا، فقد يتم تنفيذ السكربت باستخدام الشيل الافتراضي للمستخدم، والذي قد لا يكون باش، مما قد يؤدي إلى مشاكل في التنفيذ.

بشكل عام، إذا كنت تريد كتابة سكربت باش، يُفضل أن تشمل السطر "#!/bin/bash" في البداية كجزء من الطريقة الجيدة في كتابة السكربتات.

echo

في لغة الباش (Bash)، الأمر `echo` هو أحد الأوامر الأساسية التي تُستخدم لطباعة strings أو حتى المتغيرات على الشاشة.

```bash
#!/bin/bash

echo "hello"

```

Bash

```bash
# Terminal

./first-script.sh
hello

```

Bash

المتغيرات Variables

تُستخدم المتغيرات لتخزين البيانات التي يمكن الرجوع إليها في جميع أنحاء النص البرمجي. أسماء المتغيرات حساسة لحالة الأحرف Case Sensitive وعادةً ما تتكون من حروف كبيرة، حروف صغيرة، أرقام، وشرطات سفلية (_). يتم تعيين قيمة إلى متغير باستخدام الصيغة `اسم_المتغير=القيمة`. مثال:

```bash
# إنشاء المتغير
greeting="مرحبًا، العالم!"

# عرض قيمة المتغير
echo $greeting

```

Bash

اشارة الاقتباس الفردية '

- يتم استخدامها لعرض النص المتضمن فيها كما هو حرفياً.
- لا يتم عرض قيم المتغيرات فيها.

```bash
#!/bin/bash

name=Nawras

echo 'My name is $name'
```

Bash

```
```

```bash
# Terminal

./first-script.sh
My name is $name

```

Bash

اشارة الاقتباس المزدوجة “

- يتم استخدامها لعرض النص المتضمن فيها.
- يتم عرض قيم المتغييرات فيها.

```bash
#!/bin/bash

name=Nawras

echo "My name is $name"
```

Bash

```bash
# Terminal

./first-script.sh
My name is Nawras

```

Bash

التعليقات

تبدأ التعليقات في النصوص البرمجية للباش برمز # وتستمر حتى نهاية السطر. تُستخدم لتوثيق النصوص البرمجية، وتوفير شروحات أو تذكيرات لمستخدمي النص البرمجي أو المطورين. مثال:

```bash
# هذا تعليق لا يتم تنفيذه
```

Bash

التكرار Loops

تُستخدم Loops لتنفيذ مجموعة من الأوامر مرارًا وتكرارًا. يدعم باش الحلقات `for`، `while`، و `until`. مثال:

```bash
for i in {1..5}; do
    echo "الرقم : $i"
done
```

نتيجة التنفيذ Exit Status

حالة التنفيذ (Exit Status) في لغة الباش (Bash) هي قيمة تُشير إلى نتيجة تنفيذ أمر معين أو سكربت. عادةً ما يكون 0 يُعتبر رمزًا للنجاح، في حين أن أي قيمة غير صفر تُعتبر رمزًا للفشل.

بمجرد انتهاء تنفيذ الأمر أو السكربت، يتم تخزين حالة التنفيذ في متغير يُسمى "حالة الخروج" (exit status)، والذي يمكن الوصول إليه باستخدام المتغير "$?".

```bash
# Terminal

./first-script.sh

0
```

في البرمجة، يُمكن استخدام حالة التنفيذ لاتخاذ القرارات بناءً على نتيجة تنفيذ الأمر، مثل تنفيذ أوامر إضافية في حالة النجاح أو عرض رسالة خطأ في حالة الفشل.

Exit 0

تمثل قيمة 0 تنفيذ السكربت أو الأمر بشكل ناجح.

```bash
#!/bin/bash

mkdir myfolder
echo $?

```

Bash

في المثال السابق: تم إنشاء سكربت لإنشاء مجلد جديد باسم myfolder، ثم تم طباعة على الشاشة إذا تم الأمر بنجاح باستخدام معامل $?.

عند تنفيذ السكربت، كانت حالة الخروج أو exit تساوي 0 لأن تم تم فعليا إنشاء المجلد بنجاح.

Exit failure

أي رقم عدا 0 في قيمة حالة exit يعتبر فشلاً في تنفيذ السكربت أو الأمر.

```bash
#!/bin/bash

rm -r lolo
echo $?
```

Bash

```
# Terminal

./first-script.sh

rm: cannot remove 'lolo': No such file or directory
1

```

Bash

في المثال السابق: تم إنشاء سكربت لحذف مجلد لولو، ثم تم طباعة على الشاشة إذا تم الأمر بنجاح باستخدام معامل $?.

لا يوجد مجلد بهذا الاسم، ولذلك تم عرض نتيجة تنفيذ السكربت 1 أي يعني لا تساوي صفر. يعني حالة

فشل في التنفيذ.

تنفيذ كل أمر أو سكربت في لغة الباش يترتب عليه رقم خروج (Exit Code) مختلف يُمثل حالة تنفيذه. هذا الرقم يساعد في فهم نتيجة التنفيذ وتحديد ما إذا كان الأمر قد نجح أم فشل. تذكر أن القيمة الافتراضية للنجاح هي 0، أما قيم الفشل فتكون غير صفرية.

راجع قسم المعاملات الخاصة في هذا الدليل لتعلم المزيد

الجمل الشرطية Conditional Statement

في لغة الباش (Bash)، تُستخدم العبارات الشرطية (Conditional Statements) لاتخاذ القرارات بناءً على قيم معينة أو حالات مختلفة. يتم تنفيذ كود معين إذا تم تحقيق الشرط المحدد، ويتم تجاهله إذا لم يتم ذلك. إليك بعض العبارات الشرطية الأساسية في لغة الباش:

if statement

تستخدم لتنفيذ كود معين إذا تم تحقيق الشرط المحدد.

```bash
if [ condition ]; then
    # code to execute if condition is true
fi

```

Bash

```
```

```bash
#!/bin/bash

name="Ali"

if [ "$name" = "Ali" ]; then
    echo "Hello Admin"
fi

```

Bash

```bash
# terminal

./first-script.sh

Hello Admin
```

Bash

انتبه: لا تنس أن تضع فراغ بين [] عند كتابة الشرط.

if-else statement

تستخدم لتنفيذ كود معين إذا تم تحقيق الشرط، وكود مختلف إذا لم يتم ذلك.

```bash
if [ condition ]; then
    # code to execute if condition is true
else
    # code to execute if condition is false
fi

```

Bash

```bash
```

```bash
#!/bin/bash

name="Lala"

if [ "$name" = "Ali" ]; then
    echo "Hello Admin"
else
    echo "You are not the admin"
fi

```

Bash

```bash
# terminal

./first-script.sh

You are not the admin
```

Bash

elif statement

تستخدم لتحديد عدة شروط متعددة وتنفيذ كود مختلف لكل شرط.

```bash
if [ condition1 ]; then
    # code to execute if condition1 is true
elif [ condition2 ]; then
    # code to execute if condition2 is true
else
    # code to execute if all conditions are false
fi

```

Bash

```bash
#!/bin/bash

number=15

if [ $number -gt 20 ]; then
    echo "$number is greater than 20"
elif [ $number -eq 15 ]; then
    echo "$number is equal to 15"
else
    echo "$number is less than 20 and not equal to 15"
fi

```

Bash

```bash
# terminal

./first-script.sh

15 is equal to 15
```

Bash

معاملات المقارنة Comparison Operators

في لغة الباش (Bash)، تستخدم معاملات المقارنة للقيام بالمقارنات بين القيم وإرجاع قيمة صحيحة أو خاطئة بناءً على نتيجة المقارنة. إليك بعض معاملات المقارنة الأساسية في باش:

معامل المقارنة	الوصف	مثال
=	يساوي: تستخدم لـ strings	`[ "var1" = "$var2$" ]`
!=	لا يساوي: تستخدم لـ strings	`[ "var1" != "$var2$" ]`
-eq	يساوي: تستخدم لـ الأرقام	`[ "num1" -eq "$num2$" ]`
-ne	لا يساوي: تستخدم لـ الأرقام	`[ "num1" -ne "$num2$" ]`
-lt	أقل من | Less Than	`[ "num1" -lt "$num2$" ]`
-le	أقل من أو يساوي | Less Than or Equal to	`[ "num1" -le "$num2$" ]`
-gt	أكبر من | Greater Than	`[ "num1" -gt "$num2$" ]`

معامل المقارنة	الوصف	مثال
-ge	أكبر من أو يساوي | Greater than or or Equal to	`[ "num1" -ge "$num2$" ]`

جدول يوضح أهم معاملات المقارنة التي يمكن أن تستخدمها في لغة Bash

المعاملات المنطقية Logical Operator

يمكن استخدام المعاملات المنطقية في لغة Bash لتحقيق شروط معقدة أو تجميع عدة شروط معًا. هنا بعض المعاملات المنطقية الشائعة:

المعامل المنطقي	الوصف	مثال
&&	و - AND	`[ "var2" = "value2$" ] && [ "var1" = "value1$" ]`
||	أو - OR	`[ "var" = "value2$" ] \|\| [ "var" = "value1$" ]`
!	ليس - NOT	`[ "var" = "value$" ] !`

جدول يوضح أهم المعاملات المنطقية التي يمكن أن تستخدمها في لغة Bash

مثال على استخدام المعاملات المنطقية معاملات المقارنة معاً.

```bash
#!/bin/bash

num=1
num=20


# Logical AND
if [ "$num1" -gt 0 ] && [ "$num2" -gt 0 ]; then
    echo "Both numbers are greater than 0"
fi

# Logical OR
if [ "$num1" -eq 0 ] || [ "$num2" -eq 0 ]; then
    echo "At least one of the numbers is equal to 0"
fi

# Logical NOT
if ! [ "$num1" -eq "$num2" ]; then
    echo "The numbers are not equal"
fi

```

Bash

التكرار Loops

تُستخدم Loops لتنفيذ مجموعة من الأوامر مرارًا وتكرارًا. يدعم باش الحلقات `for`، `while` و `until`. مثال:

```bash
for i in {1..5}; do
    echo "الرقم: $i"
done
```

Bash

هذا القسم قيد الكتابة. سيتم اضافة هذا القسم لاحقاً.

الدوال Functions

تسمح الدوال في باش بتجميع الأوامر معًا وإعادة استخدامها في جميع أنحاء النص البرمجي. يتم تعريف الوظائف باستخدام الكلمة المفتاحية `function` أو ببساطة تقديم اسم الدالة تلاه قوسان. مثال:

```bash
# تعريف الدالة
my_function() {
    echo "مرحبًا من داخل الدالة!"
}

# استدعاء الدالة
my_function

```

Bash

```bash
# Terminal

./first-script.sh

مرحبًا من داخل الدالة!
```

Bash

المعاملات Parameters

المعاملات أو Parameters هي المتغيرات التي يتم استخدامها داخل الدوال أو Functions.

```bash
# تعريف الدالة
my_function() {
    echo $1 # paramater 1
    echo $2 # parameter 2
}

# استدعاء الدالة
my_function

```

Bash

المعاملات المُمررة Arguments

هي القيم التي تقوم بتمريرها للدالة Function عند تنفيذها

```bash
hello # argument 1
User # argument 2

# Terminal

./first-script.sh hello User

hello
User


```

Bash

ما هي Paramaters أو المعاملات؟ ما هي Arguments أو المعاملات الممررة؟ وما الفرق بينها؟ راجع مقتطف الفرق بين Parameters و Arguments لتعلم المزيد.

المعاملات في لغة Bash

المعاملات الموضعية | Positional Parameters

المعاملات الموضعية هي متغيرات خاصة في برمجة الشل (Bash) تستخدم للوصول إلى القيم التي يتم تمريرها إلى نص برنامج أو دالة. يمكن استخدام هذه المتغيرات للتعامل مع المدخلات المختلفة التي يتم تمريرها إلى البرنامج أو الدالة.

```bash
./first-script.sh hello you

./first-script.sh
hello
you
```

Bash

```bash
# File Name : first-script.sh

echo $0
echo $1
echo $2
```

اسم المعامل	توضيح
$0	تحمل قيمة اسم ملف سكربت الذي يحتوي على الكود
$1	تحمل قيمة أول arg يتم تزويده عند تنفيذ ملف السكربت
$2	تحمل قيمة ثاني arg يتم تزويده عند تنفيذ ملف السكربت
$3	تحمل قيمة ثالث arg يتم تزويده عند تنفيذ ملف السكربت
$4	تحمل قيمة رابع arg يتم تزويده عند تنفيذ ملف السكربت
$5	تحمل قيمة خامس arg يتم تزويده عند تنفيذ ملف السكربت
$6	تحمل قيمة سادس arg يتم تزويده عند تنفيذ ملف السكربت
$7	تحمل قيمة سابع arg يتم تزويده عند تنفيذ ملف السكربت
$8	تحمل قيمة ثامن arg يتم تزويده عند تنفيذ ملف السكربت
$9	تحمل قيمة تاسع arg يتم تزويده عند تنفيذ ملف السكربت

جدول يوضح قيم المعاملات الموضعية في لغة Bash

المعاملات الخاصة | Special Parameters

المعاملات الخاصة في لغة Bash تُستخدم للتفاعل مع البيانات والمتغيرات التي يتم تمريرها إلى السكربتات أو الدوال عند تنفيذها. تُسهّل هذه المعاملات التحكم في السلوك وتنفيذ العمليات بناءً على البيانات المُمررة. الجدول الآتي يوضح بعضها:

اسم المعامل	توضيح
$@	يقوم بوضع كل المعاملات المُمررة ككقائمة أو array
$#	عدد المعاملات المُمررة

اسم المعامل	توضيح
$*	يقوم بوضع كل المعاملات المُمررة ككقائمة أو array
$?	يمثل حالة exit لآخر أمر تم تنفيذه
$_	يمثل المعامل الممرر الأخير الذي تم استخدامه في الأمر السابق

جدول يوضح أهم المعاملات الخاصة في لغة باش

المعامل الخاص $#

```bash
./first-script.sh hello you 5

./first-script.sh
Number of passed Args: 3
```

Bash

```bash
# File Name : first-script.sh

echo "Number of passed Args: $#"
```

Bash

المعاملات الخاصة $@ و $*

```bash
./first-script.sh hello you 5

./first-script.sh
All args are: hello you 5
All args are: hello you 5
```

Bash

```bash
# File Name : first-script.sh

echo "All args are: $*"
echo "All args are: $@"
```

```
Bash
```

الفرق بين معامل @$ و *$ في لغة باش

> هذا القسم قيد الكتابة. سيتم اضافة هذا القسم لاحقاً.

المعامل الخاص ?$

```bash
# File Name : first-script.sh

mkdir folder
echo "The exit status is $?"
```

Bash

```
./first-script.sh

The exit status is 0

```

Bash

في المثال السابق: تم إنشاء سكربت لإنشاء مجلد جديد باسم folder، ثم تم طباعة على الشاشة إذا تم الأمر بنجاح باستخدام معامل ?$.

عند تنفيذ السكربت، كانت حالة الخروج أو exit تساوي 0 لأن تم تم فعليا إنشاء المجلد بنجاح.

المعامل الخاص _$

عند تنفيذ سطر في الموجه، قد يكون لديك مخرجات أو نتائج تم إنشاؤها. في حالة استخدام $_، يُرجع النص الذي تم استخدامه كآخر معامل مُمرر في الأمر الذي تم تنفيذه. هذا يمكن أن يكون مفيدًا عندما تحتاج إلى استخدام المعامل الممرر الأخير في أمر جديد.

```
```

```bash
echo "Hello, World"
Hello, World

echo "$_"
Hello, World

```

Bash

```bash
ls
file1.txt  file2.txt  file3.txt

mv file3.txt new_file.txt

echo "تم نقل الملف إلى $_"
تم نقل الملف إلى new_file.txt


```

Bash

عزيزي المتابع: كتابة سكربتات باش قد يكون فيه بعض من التعقيد. سأقوم بتحديث هذا الدليل بشكل دوري حتى اضيف أمثلة أكثر وتوضيحات أكثر حول جوانب هذه اللغة. تابع مسار سكربتات باش المجاني والمتاح للجميع لتعلم كتابة الأكواد في لغة Bash بطريقة أقرب للواقع أكثر.

مشاريع عملية وشرح المفاهيم الأساسية. نولي اهتمامًا كبيرًا أيضًا للجانب الإنساني والنفسي للمبرمجين.

الموضوعات الرئيسية

- Git
- أدوات ونصائح
- أقوال وحكم
- أوامر لينكس
- السيرفرات ولينكس
- تطوير الويب
- دروس مباشرة
- عام
- قواعد البيانات
- ووردبريس

الدليل الشامل

لغة Bash

روابط مفيدة

صفحة تمارين جافاسكربت

أرشيف مدونة تعلم مع ناو (غير محدث)

قناة اليوتيوب

رابط مستودع الكود

ويكي إدارة المواقع (غير محدث)

اسئلة تقنية عامة (LWN Quiz)

التواصل

إذا كنت بحاجة إلى طلب خدمة أو لديك أي استفسار، يُرجى مراسلتي عبر البريد الإلكتروني. سأكون سعيدة بمساعدتك فيما تحتاجه.البريد الإلكتروني:

mail@LearnWithNaw.net