الأساسي في البيرل

بن العيد

27-10-2008

الدرس الأول و الثاني

الفهرس

مقدمة

يتكرر التساؤل حول جدوى تعلم لغة البرمجة البيرل و إستخداماتها لذلك ستكون إجابتي على هذه التساؤلات بمثابة مقدمة عامة لهذه السلسلة من الدروس.
كان تعلمي للبيرل نتيجة لطبيعة عملي السابق، حيث كنت مطالب بتنفيذ تعليمات متوالية على الواجهة console في نظام التشغيل اليونيكس, بعض هذه التعليمات يتطلب الكثير من الوقت يفوق بعضها 12 ساعة و بديهي أن تنفيذ أمر كل مرة و إنتظار نهايته للمرور إلى ما يليه أمر يستدعي السخرية. لذلك فإن ال scripting يعد الحل الأمثل إلا أن ال batch و ال ksh لم يكونا الحلين الأمثلين, فكانت هناك حاجة للغة أخرى تكون سلسلة أكثر و أبسط و هو ما توفره لغة البرمجة البيرل.
لم تكن مهمتي صعبة إنما روتينية و تطلب الكثير و الكثير من التركيز لأن أي خطأ في إحدى المراحل سيغير النتيجة النهائية التي عادة ما تكون متوقعة; و هي عدد المكونات التي تغيرة في نهاية ترجمة شفرة نظام التشغيل. ساعدني مجددا البيرل في إجتناب أخطاء المنجرة عن قلة التركيز عبر تحويل الأوامر و التعديلات اليدوية إلى شفرة بيرل تنفذ المطلوب و تتأكد من صحة النتيجة.
مسألة أخرى هامة جدا و حلها البيرل متمثلة في رفع نسخة من نظام التشغيل في الموقع الخاص للشركة نهاية عملية الترجمة و إبلاغ الروس أوتوماتيكيا بوجود النسخة, بإستخدم المكتبة Telnet للربط بالمستخدم و المكتبة FTP لنقل الملفات. فبينما نحن نحظر للذهاب للفراش يكون الروس قد حمّلوا النسخة و شرعوا في تجربة الإصدار الموجود من نظام التشغيل و في الصباح نجد تقرير الإختبار في الإنتظار ... و لكم تصور كم من الوقت تم ربحه.

مجال آخر هام جدا يتخصص فيه البيرل و بدون منازع يتمثل في القرصنة. يوفر البيرل العديد من المكتبات التي توفر إمكانية التعامل مع الشبكات و بطريقة يسيرة مما جعلها لغه القراصنة المحبذة. يعد الكتاب Breaking into computer networks from the Internet ل Roelof Temmingh أبرز داعم لما ذكرت.

حاليا أستخدم مجددا البيرل لتقيم جودة الشفرة التي طورها فريق العمل المنتمي له إثر فرض إستخدام سكريبت بيرل لتغطية نقائص الأداة Splint[1]، فغير منطقي البحث اليدوي في ملفات تحتوي الآلاف من الأسطر عن متغيرات مسمات بحرف واحد أو إذا إستخدم أحد المطورين تعليمات مثل ... goto, #endif فيسر هذا السكريبت عملية البحث الآلي و إكتشاف التجاوزات في قواعد كتابة الشفرة.

[1]Splint1 is a tool for statically checking C programs for security vulnerabilities and programming mistakes.

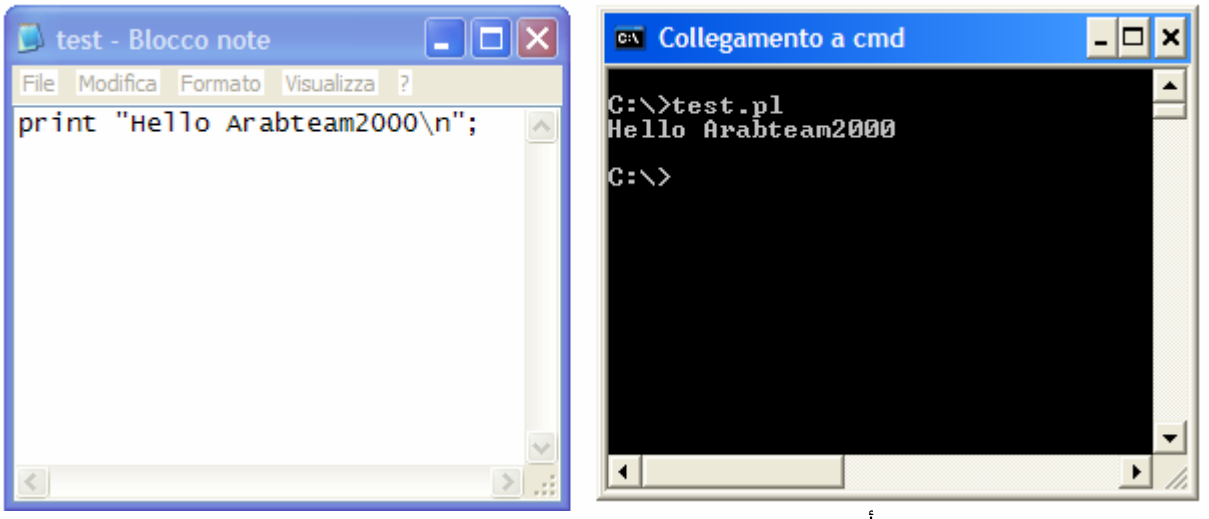
II المتغيرات, الجمل الشرطية و الحلقات

1. مقدمة

نظرا لأهمية لغة البرمجة البيرل في تطوير البرمجيات و ما تتسم به من ثراء معرفي أشارككم في هذه الأسطر خبرتي المتواضعة في هذا المجال. هذا الدرس، إن صحت تسميته كذلك، يخص بالأساس الجانب التطبيقي في أساسيات البيرل و هو موجه للمبتدئين و إن شاء الله في الدروس القادمة سيكون هناك ما يستفيد به حتى ذوي الخبرة.

اللوازم

تتمثل لوازم البرمجة بلغة البيرل أساسا في المترجم ActivePerl [1] و تتعدد إصداراته و فق نظام التشغيل و المعالج أيضا. إذا كان نظام التشغيل الذي تستعمله هو الويندوز فيستا فبإمكانك إستعمال هذا الإصدار [2]. لكتابة الشفرة يكفي إستخدام محرر النصوص (الصورة 1.ب) و للترجمة فعليك فتح ال cmd و كتابة إسم السكريبة من مسار تواجده أو ينبغي كتابة المسار كامل (الصورة 1.أ).



ب أ

الصورة رقم1: اللوازم.

2. المتغيرات

تعتمد لغة البرمجة بيرل على الرمز $ لتعريف المتغيرات و تشترط إستخدام المتغيرات وفق ترتيب تعريفها، بمعنى أن أي إستخدام لمتغير لم يتم تعريفه مسبقا أو مطلقا يتم تجاهله في عملية الترجمة.

لتعريف المتغير بشكل عام يستخدم في أي جزء من الشفرة خاصة عند إستخدام الدوال, ينبغي سبق تعريفه بالكلمة المحجوزة my.

سنجرب جمع متغيرين و إظهار النتيجة على الشاشة كما هو في المثال التالي:

```
$a      =10;
$b      =20;
$sum    =0;

print "a+b=$a +$b\n";
$sum =$a + $b;
print "a+b=$sum";
#####
print "a+b=sum";
print "a+b=$a +$c\n";
```

نلاحظ أن في تعليمة الطباعة الأولى تم إدراج عملية الجمع مباشرة داخل النص الذي سيتم إظهاره على الشاشة، في حين أن في التعليمة الثانية تم إدراج ناتج الجمع في النص المُخرج. أدرجت تعليمتين إضافيتين لطباعة عملية الجمع و لكن بتغيرات طفيفة متمثلة في حذف الرمز $ الذي سبق متغير نتيجة الجمع و إدراج المتغير $c.

بعد حفظ الشفرة السابقة في ملف test.pl في المسار c:\, في ما يلي نتيجة الترجمة:

```
C:\> perl test.pl
a+b=10 +20
a+b=30
a+b=sum
a+b=10 +
C:\> test.pl
a+b=10 +20
a+b=30
a+b=sum
a+b=10 +
```

عند الترجمة ينبغي تحديد المسار الكامل للملف pl. إذا لم يكن موجد في المسار الحالي الذي يمكن معرفته بتنفيذ الأمر echo %CD%. لترجمة الشفرة يمكن كتابة إسم الملف مع الإمتداد مسبوقا بالتعليمة perl أو بدونها لأن إثر تنصيب البيرل يتم إضافة مكتباته مع قائمة المسارات التي يعتمدها نظام التشغيل و يمكن مطالعة ذلك بإستخدام الأمر %PATH%. في حين أن إدراج التعليمة perl أمر إجباري إذا كان نظام التشغيل المعتمد اليونيكس و فروعه.

نعود لما طبعته الشفرة من مخرجات و التي تمتد على أربعة أسطر بنفس عدد تعليمات الطباعة كما أردنا. في السطر الأول من النص المُخرج تم فقط تعويض المتغيرين بقيمتهما دون القيام بعملية الجمع و السبب في ذلك أن علامة الجمع كانت بمثابة حرف كباقي الحروف المكونة للجملة التي يراد طباعتها على الشاشة، في حين أن في السطر الثاني تم إظهار النتيجة المنتظرة لأن عملية الجمع تمت خارج تعليمة الطباعة.

الغريب أن في السطر الثالث تم إرجاع إسم المتغير عوض إبراز قيمته و السبب في ذلك ما ذُكر أنفا من ضرورة سبق أسماء المتغيرات بالرمز $. بالسطر الأخير من المخرجات نتذكر دائما أن المترجم يتجاهل أي إستخدام لمتغير غير معرف و قراءة قيمة متغير لم يتم إسناده قيمة أولية.

ملاحظات:

- إذا أردت إدراج تعليقات في الشفرة فينبغي بدأ كل السطر بالرمز # سوى كان التعليق في سطر واحد أو يمتد على عدة أسطر.
- دالة الطباعة print تماثل printf في السي و نظيراتها في باقي لغات البرمجة حيث يمكن التحكم في شكل النص المخرج و توزعه بإستخدام الرموز \a, \t, \n و غيرها.

3. الجمل الشرطية و الحلقات

لا شكل أن الحلقات و الجمل الشرطية تمثل العمود الفقري لأي لغة برمجة بما فيها البيرل، حيث تمكن المطور من معالجة كافة الإستثناآت التي قد تحدث و تجنب التكرار الآلي للتعليمات.

1.3. الجمل الشرطية

قبل الشروع في شرح إستخدام الجمل الشرطية، أذكر بمجموعة المعاملات التي يمكن إستخدامها في عمليات المقارنة بين المتغيرات. تستخدم الرموز المذكورة في الجدول أساسا في المقارنة بين المتغيرات النصية مع إمكانية إستخدامها مع باقي المتغيرات العددية.

الرمز	المرادف
eq	==
ne	!=
le	<=
ge	>=

في ما يلي مثال لمقارنة بين نصين لاتينيّن بإستخدام كافة علامات المقارنة، و الغاية منه أولا تجربة هذه المعاملات[2] و فهم أساس المقارنة بين النصوص ثانيا.

```
$var1= "AZX";
$var2= "BAX";

if($var1 ne $var2)
{ print "var1 isn't equal to var2.\n";}
if ($var1 eq $var2)
{ print "var1 is equal to var2.\n";}
if ($var1 ge $var2)
{ print "var1 is greater or equal to var2.\n";}
if ($var1 le $var2)
{print "var1 is less or equal to var2.\n";}
```

نتيجة الترجمة كانت كالآتي:

```
var1 isn't equal to var2.
var1 is less or equal to var2.
```

[2] المعاملات: Operators.

جلي أن المتغيران غير متعادلان و لكن لماذا المتغير الثاني أكبر من الأول؟
المقارنة بين النصوص ترتكز على الترتيب ASCII للحروف و الحرف الأسبق من غيره يعد الأكبر, أما داخل النص في حد ذاته فهناك ترتيب تفاضلي لمدى أهمية الحروف أيضا حسب ترتيبه في الجملة. لذلك فإن الحرف الأول هو الأكبر. لو إطلعنا على النصين المعنيين فسنلاحظ أن النص الأول يبدأ بالحرف A في حين أن الثاني يبدأ بالحرف B و هو أكبر, و رغم عكس الآية و وضع حرف ثاني في النص الأول أكبر من الثاني فإن النتيجة لم تتغير نظرا لأهمية ترتيب الحروف. بالإمكان الآن إعادة التجربة بإستخدام الرموز الرياضية للمقارنة و ستكون النتيجة ذاتها.
تستخدام else و elsif في الجمل الشرطية و يبقى إستخدامها مماثل لبقية لغات البرمجة.

2.3. الحلقات

سأكتفي في هذا الجزء من الدرس بسرد أمثلة بسيطة و سريعة حول الحلقات إلا حين التقديم للمصفوفات حيث سيتم الرجوع للحلقات بأكثر تفاصيل.

```
for $i (1 , 2 , 3 , 4 , 5)
{ print "$i "; }

print "\n";

for $i (1 .. 5)
{ print "$ i"; }

print "\n";
$i=0;
while($i<=5)
 { print "$i ";
  $i++;
 };
```

نتيجة الترجمة كانت كالآتي:

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

و هو ما يثبت أن للحلقات الثلاثة نفس الوضيفة.

4. خاتمة

هذا الدرس بمثابة مقدمة عامة حول البرمجة بالبيرل و سرد لأبجدياتها و ستكون هناك سلسلة متكاملة من الدروس إن شاء الله.

III معالجة النصوص

1. مقدمة

في هذا الدرس سنهتم بمعالجة النصوص و ما يتعلق بها من ربط و تقسيم و بحث داخلها و غير ذلك من المهام التي يمكن للمطور إحتياجها.

2. معالجة النصوص

المقصود بمعالجة النصوص ذكر العمليات الأساسية التي قد يحتاجها المطور من جمع، قسمة والبحث في النصوص. هاته المعالجة تحمل طابع تطبيقي عملي أكثر من كونه نظري تحليلي.

- جمع النصوص

تتعدد الطرق لجمع النصوص و الكلمات و يبقى أبرزها عبر إستخدام الدالة join. بالإضافة إلى هذه الأخيرة يمكن إستخدام المعامل "." أو بكل بساطة الجمع الضمني داخل نص آخر. المثال التالي يسد كيفية إستخدام الطرق الثلاثة و يبقى للمطور إستخدام الطريقة الأنسب له.

```
$String1="This is ";
$String2="a tutorial";

$String3="$String1$String2\n";
#Show output string and reset content
print $String3;
$String3="";

$String3.= $String1;
$String3.= $String2;
#Show output string and reset content
print $String3;
$String3="";

$String3= join($String3, $String1 ,  $String2);
print "\n$String3";
```

نتيجة ترجمة الشفرة كانت تكرار نفس النص ثلاثة مرات. النص المُخرج في كل مرة هو جمع بين النصين المختارين مسبقا و لكن الربط بينهما كان بطرق مختلفة.

```
This is a tutorial
This is a tutorial
This is a tutorial
```

كما نحتاج لربط النصوص نحتاج أيضا لتقسيمها، و هو ما سيكون هدف الفقرة الموالية من هذا الدرس.

- تقسيم النصوص

سنكتفي في هذا الجزء بإبراز كيفية إستخدام الدالة Split. تعتمد هذه الدالة أساسا على مدخلين وهما النص المدخل و النص الذي سيعتمد عليه في التقسيم. أما مخرجاتها فهي مصفوفة متكونة من أجزاء النص المقسم.

لم يتم الحديث بعد عن المصفوفات في هذا الدرس أو سابقه، لذلك أُذكر أن هذا النوع من المتغيرات يتميز بحجزه للرمز @.

```
$InputString="This is a tutorial";
$Separator=" ";
$count=0;

@Array = split($Separator, $InputString);
$count =@Array;

for($k=0; $k<$count; $k++)
 {
  print "$k)  @Array[$k]\n";
 }
```

إعتمدنا في عملية التقسيم على الفراغات بين الكلمات إن وجدت. إثر التقسيم و خزن أجزاء النص المقسم في مصفوفة، تم إستخدام دالة الطباعة لإظهار محتوياتها فكانت النتيجة كما يلي:

```
0) This
1) is
2) a
3) tutorial
```

أُذكر أنّ في عملية التقسيم يتم حذف الحرف أو النص المستخدم في عملية التقسيم. مثال على ذلك تبرزه نتيجة ترجمة الشفرة السابقة بإستخدام الحرف s عوض عن فراغ:

```
0) Thi
1) i
2) a tutorial
```

- البحث داخل النص

قد يحتاج المطور للبحث داخل نص معين ما إن كان يتطابق مع مواصفات معينة أم لا. للإستجابة لهذه الحاجة تتوفر عدة دوال و سبل للبحث أبرزها إستخدام الدالة index. هذه الدالة ترجع 1- إذا لم يوجد النص المعني في النص المدخل و غير ذلك إذا وُجد.

```
$InputString="This is a tutorial";

if(index($InputString,"H")==-1)
{
  print "Not Found";
}
```

في الشفرة السابقة بحث ما إن كان الحرف H من بين الحروف المكونة للنص المعطى أم لا مع طباعة نص مأكد بعدم وجوده. يقابل إستخدام الدالة index المعامل، فيصبح المثال السابق كما يلي:

```
$InputString="This is a tutorial";
$String1="H";
$String2="tu";

if($InputString !~ $String1)
 {
  print "\"$String1\" was not found in \"$InputString\".\n";
 }
if ($InputString=~$String2)
 {
  print "\"$String2\" was found in \"$InputString\".";
 }
```

كالعادة ننتقل إلى نتيجة الترجمة و نلاحظ في هذا المثال أنه تم إستخدام ثلاثة متغيرات؛ سنقوم بالتأكد من و جود قيمتا إثنان منهما في قيمة المتغير الثالث.

```
"H" was not found in "This is a tutorial".
"tu" was found in "This is a tutorial".
```

بإمكان المطور البحث عن أكثر من نص في نص آخر, و للقيام بذلك يكفي إستخدام المعامل "|". مثال في ذلك:

```
if ($InputString=~"$String2|$String1") { }
```

3. خاتمة

أختم هذا الدرس بالتذكير أن لغة البيرل ثرية جدا و لا يكفي الصفحات و الصفحات لكشف خباياها وميزاتها، و بما أننا سنركز في هذه الدروس على أساسياتها فسأكتفي بسرد النقاط الضرورية وبشكل عملي.

يتبع ...

الروابط

[1] http://www.activestate.com/store/download_file.aspx?binGUID=d8129061-73ad-4361-b8a0-20ce0195095

[2] http://www.softpedia.com/get/Programming/Coding-languages-Compilers/ActivePerl.shtml