LAB: SHELL PROGRAMMING

2.1 Biến shell

2.1.1 Tạo và sử dụng biến: =, $, set, unset

Biến trong shell không cần được khai báo trước và cũng không cần chỉ rõ kiểu dữ liệu. Khi ta sử dụng biến lần đầu tiên có nghĩa là ta định nghĩa biến đó. Tên biến là một dãy ký tự gồm có các chữ cái, chữ số hoặc ký tự gạch dưới. Chữ số không được đứng đầu tên biến, tên biến không được chứa khoảng trống hay bất kỳ một ký tự khác.

a. Gán giá trị cho biến: =

Để gán giá trị cho biến, ta dùng toán tử gán "=" Chú ý rằng trước và sau ký tự "=" không được có khoảng trắng.

Ví dụ 1:

```
$a=Hello
```

Khi đó biến *a* được tạo và nhận giá trị là "*Hello*".

Bạn cũng có thể thay đổi kiểu giá trị biến bất kỳ khi nào.

b. Lấy giá trị của một biến: $

Để tham chiếu đến giá trị một biến, bạn dùng toán tử "$" ở phía trước tên biến.

Ví dụ 1: Dùng câu lệnh *echo* để hiển thị giá trị của một biến đã được gán giá trị

```
$so_pi=3.141592654
$echo $so_pi
```

2.1.2 Các dấu bao xâu

Giá trị mà bạn gán cho biến có thể là một dãy ký tự tùy ý. Các ký tự này có thể là do bạn nhập vào hoặc là kết quả của một lệnh nào đó. Trong nhiều trường hợp, bạn cần phải bao đóng xâu ký tự đó bằng nháy kép, nháy đơn, nháy đơn ngược (**" "**, **' '**, **` `**).

- *Bao xâu bởi ' '*: Nội dung xâu kết quả sẽ chính là dãy ký tự có trong xâu được bao, bất kể trong đó là ký tự gì.

  Ví dụ:

  ```
  $mystr='Hello World!'
  $echo $mystr
  ```

  → Màn hình hiển thị: **`Hello World!`**

- *Bao xâu bởi " "*: Nếu trong xâu có dùng biến thì giá trị của biến đó sẽ hiện ra trong xâu kết quả. Trong trường hợp bạn muốn dùng dấu $ trong xâu nhưng không với ý nghĩa lấy giá trị biến, bạn phải thêm dấu \ trước ký tự $.

  Ví dụ:

  ```
  $monhoc='OSN'
  $echo "Chao mung ban den voi lop hoc $monhoc"
  ```

  → Màn hình hiển thị: **`Chao mung ban den voi lop hoc OSN`**

- *Bao xâu bởi ``*: Trong xâu phải là một câu lệnh. Xâu kết quả chính là kết quả của câu lệnh đó.

  Ví dụ:

  ```
  $timer=`date`
  ```

```
$echo $timer
```
→ Màn hình hiển thị thời gian hiện tại.

2.2 Một số câu lệnh lập trình thông dụng

2.2.1 Nhập và xuất: Lệnh read và echo

Lệnh *echo* dùng để xuất dữ liệu ra màn hình. Sau khi in dữ liệu, *echo* sẽ tự động xuống dòng. Nếu không muốn xuống dòng, bạn cần đưa thêm tùy chọn *-n*. Để nhập dữ liệu từ bàn phím ta dùng lệnh *read*. Có thể kết hợp *echo* và *read* để tạo sự tương tác giữa người dùng và hệ thống trong các script.

Ví dụ:

```
$echo “Nhap mot so nguyen : ”
$read num
88
$echo “Gia tri vua nhap la: $num”
Gia tri vua nhap la: 88
```

2.2.2 Tính toán số học: Lệnh let

Lệnh *let* cho phép thực hiện các tính toán số học. Bạn có thể dùng *let* để so sánh hoặc cộng, trừ, nhân, chia hai số. Cú pháp của *let* như sau:

```
$let giá_trị_1 toán_tử giá_trị_2
```

Toán tử	Phép toán
+	Cộng
-	Trừ
*	Nhân
/	Chia
**	Lũy thừa
%	Lấy dư (modulo/mod)
+=	Cộng bằng: tăng giá trị biến thêm 1 hằng số
-=	Trừ-bằng: giảm giá trị biến đi một hằng số
*=	Thêm-bằng: thêm giá trị biến một hằng số lần
/=	Giảm-bằng: giảm giá trị biến đi một hằng số lần
%=	Dư-bằng: phần dư của phép toán chia biến cho một hằng số
Các toán tử số học	

Ví dụ 1:

```
$let 8+8
```
→ giá trị trả về sẽ là 16

Còn một cách viết biểu thức tính toán khác tương đương với lệnh *let*, đó là sử dụng cặp đôi dấu ngoặc đơn dạng: **$((biểu_thức))**. Ví dụ:

$let “sum = 2 + 3”** tương đương với **$((sum=2+3))

Ví dụ 4:

```
$num1=10
$((num1/=3))
$echo $num1
```

3

2.2.3 So sánh và kiểm tra

Ví dụ 3:

$[$num –eq 0]

→ tương đương : `$test $num –eq 0`

$[$mystr="hello"]

→ tương đương : `$test $mystr="hello"`

Ở ví dụ này đã sử dụng toán tử so sánh số nguyên *-eq* và toán tử so sánh xâu = (cả hai toán tử đều có ý nghĩa so sánh bằng). Dưới đây là các toán tử so sánh thông dụng:

Toán tử	Ý nghĩa so sánh
-eq	Bằng
-ne	Không bằng (khác)
-gt	Lớn hơn
-ge	Lớn hơn hoặc bằng
-lt	Nhỏ hơn
-le	Nhỏ hơn hoặc bằng
Các toán tử so sánh số nguyên	

Toán tử	Ý nghĩa so sánh
=	Bằng
!=	Không bằng
-z	Xâu rỗng (độ dài bằng 0)
-n	Xâu khác rỗng
Các toán tử so sánh xâu	

2.3 Shell script

2.3.1 Tạo và thực thi script

Với những công việc phức tạp, các lệnh shell thường được đưa vào trong một file chương trình gọi là kịch bản shell (shell script). Khi ta thực thi một shell script thì các câu lệnh trong đó sẽ lần lượt được thực hiện. Sau khi soạn thảo xong script, có 2 cách để thực thi script:

Cách 1:

Bước đầu tiên bạn phải gán quyền thực thi (executable) cho script đó bằng lệnh *chmod* như sau:

$chmod u+x tên_script

Sau đó có thể thực thi script bằng cách:

$./tên_script

Cách 2: Nếu không muốn gán quyền thực thi cho script, bạn có thể chạy trực tiếp script bằng lệnh *sh* như sau:

$sh tên_script

Ví dụ: viết một script có tên "*tinhtoan.sh*" đáp ứng các yêu cầu sau: mời người dùng nhập vào hai số nguyên, sau đó tính toán và đưa ra màn hình *tổng, hiệu, tích, thương* của hai số nguyên vừa nhập. Lời giải của ví dụ này gồm các bước dưới đây:
Bước 1: tạo mới và vào màn hình soạn thảo file "*tinhtoan.sh*":
Bước 2: soạn nội dung file "*tinhtoan.sh*" như sau:

```
echo -n "Nhap so nguyen thu nhat : "
read so1
echo -n "Nhap so nguyen thu hai  : "
read so2
let "tong = so1 + so2"
let "hieu = so1 - so2"
let "tich = so1 * so2"
let "thuong = so1 / so2"
echo "Tong cua $so1 va $so2 la          : $tong"
echo "Hieu cua $so1 va $so2 la          : $hieu"
echo "Tich cua $so1 va $so2 la          : $tich"
echo "Thuong cua $so1 va $so2 la        : $thuong"
```

Bước 3: Lưu nội dung vừa soạn thảo.
Bước 4: Gán quyền thực thi cho "tinhtoan.sh" bằng lệnh *chmod*:

```
$chmod u+x tinhtoan.sh
```

Bước 5: Thực thi file "tinhtoan.sh":

```
$./tinhtoan.sh
```

2.3.2 Các cấu trúc điều khiển

a. Cấu trúc điều kiện if

Cú pháp:

```
if lệnh_điều_kiện
     then
          lệnh_thực_hiện
     else
          lệnh_thực_hiện
fi
```

b. Cấu trúc điều kiện case

Cú pháp:

```
case xâu_ký_tự in
     mẫu_1)
          lệnh_thực_hiện
          ;;
     mẫu 2)
          lệnh_thực_hiện
          ;;
     …
     *)
          lệnh_thực_hiện
esac
```

c. Cấu trúc lặp while
Cú pháp:
while lệnh_điều_kiện
do
lệnh_thực_hiện
done

d. Cấu trúc lặp for
Cú pháp:
```
for biến in danh_sách_giá_trị
do
    lệnh_thực_hiện
done
```

2.4 Bài tập

Bài tập 1:

- Thực hiện các câu lệnh cần thiết để thực hiện những công việc sau: Gán giá trị 2 biến: **subject=OSN**, **name=tenban**, sau đó đưa ra màn hình dòng chữ:
  Bay gio la: (a). Ban ten la: (b). Ban dang hoc mon: (c)
  Với (a) là thời gian hiện hành, (b) là giá trị *tenban* gán cho biến *name*, (c) là giá trị *OSN* đã gán cho biến *subject*.
  Ví dụ: **Bay gio la: Tue May 27 10:14:52 ICT 2008. Ban ten la: Hoang Linh. Ban dang hoc mon OSN**
  Gợi ý :
  ```
  subject="OS&N"
  name="Hoang Linh"
  echo "Bay gio la: `date`. Ban ten la $name. Ban dang hoc mon $subject"
  ```
- Tập hợp các thao tác trên vào một tệp rồi gán quyền *executable* cho tệp đó, rồi thực thi tệp để cho ra kết quả như trên.

Bài tập 2:

- Tạo một tệp mới có tên *time.dat* có nội dung như sau:
  ```
  date
  ```
- Thử thực thi tệp *time.dat* bằng cách sau:
  ```
  ./time.dat
  ```
- Quan sát kết quả, nhận xét ý nghĩa thông báo xuất hiện (→ "*không được phép thực hiện...*")
- Gán quyền thực thi (*executable*) cho tệp time.dat
  Gợi ý: **chmod u+x time.dat** (lưu ý đọc lại nội dung Lab 1 để hiểu rõ tại sao *nên gán quyền thực thi chỉ cho đối tượng **u***)
- Thực hiện lại lệnh: **./time.dat**

Bài tập 3: Kết hợp lệnh if và while

Giải thích:

- Lệnh *clear* có tác dụng xóa màn hình tương tự như lệnh cls trong hệ điều hành Windows.

- Với câu lệnh điều kiện *if* trong trường hợp có nhiều hơn hai lựa chọn, bạn có thể sử dụng rẽ nhánh *elif*.

Yêu cầu: Viết một script cho phép người dùng kiểm tra liên tiếp các số nguyên nhập vào từ bàn phím để đưa ra màn hình thông báo số đó là: số âm, số dương, số không. Sau mỗi lần nhập số và in ra thông báo thì sẽ xóa màn hình và hỏi người dùng hãy lựa chọn (c/k) để có/không tiếp tục chương trình. Nếu chọn c thì lặp lại thao tác ở trên, nếu chọn k thì đưa ra thông báo "Kết thúc".

Lời giải: nội dung file "*baitap3.sh*":

```
tieptuc=y
while [ $tieptuc = y ]
do
    clear
    echo -n "Nhap mot so nguyen : "
    read songuyen
    if [ $songuyen -lt 0 ]; then
        echo "So am"
    elif [ $songuyen -gt 0 ]; then
        echo "So duong"
    else
        echo "So khong"
    fi
    echo -n "Co tiep tuc khong (y/n)?: "
    read tieptuc
done
echo "Ket thuc."
```

Bài tập 4: Sử dụng lệnh case để viết script hiển thị ngày, tháng, năm

Giải thích: tương tự như cách gọi lệnh trong các ngôn ngữ lập trình khác, một shell script có thể nhận đối số từ lời gọi lệnh. Khi gọi script, các đối số được nhập vào sau tên của script. Đối số của script được tham chiếu qua *biến đối số* (argument variable). Biến đối số gồm toán tử $ và số thứ tự của đối số trong câu lệnh. Đối số thứ nhất được tham chiếu bởi $1, đối số thứ hai được tham chiếu bởi $2… Biến đối số $0 chứa tên của script được gọi. Biến đối số $# chứa số đối số mà người dùng đưa vào. Biến đối số là các biến chỉ đọc (read-only), không thể thay đổi giá trị của các biến này. Giá trị của các biến đối số được đặt thông qua lệnh *set*.

Yêu cầu: Viết một script cho phép hiển thị ra màn hình thông tin về thời gian đầy đủ tiếng Việt (không dấu) theo dạng:

```
Hom nay la Thu Sau, ngay 18 thang 1, nam 2008
Thoi gian hien tai la 16:40 :49
```

Lời giải: Như đã đề tập đến ở phần trước của tài liệu, lệnh *date* dùng để hiển thị thông tin thời gian hiện tại. Kết quả của câu lệnh date có thể như sau :

```
$date
Sun Jan 27 22:06:45 ICT 2008
```

Như vậy có thể thấy tương ứng với lệnh date sẽ có 6 biến đối số. Sử dụng lệnh set:

```
$set `date`
```

```
echo "\$1: $1; \$2: $2; \$3: $3; \$4: $4; \$5: $5; \$6: $6"
```

Kết quả trên màn hình sẽ là:

```
$1: Sun; $2: Jan; $3: 27; $4: 22:09:28; $5: ICT; $6: 2008
```

Nội dung file "*baitap4.sh*" có thể như sau:

```
set `date`
case $1 in
    Mon)
        Thu=Hai
        ;;
    Tue)
        Thu=Ba
        ;;
    Wed)
        Thu=Tu
        ;;
    Thu)
        Thu=Nam
        ;;
    Fri)
        Thu=Sau
        ;;
    Sat)
        Thu=Bay
        ;;
    *)
        Thu="Chu nhat"
        ;;
esac
case $2 in
    Jan)
        Thang=1
        ;;
    Feb)
        Thang=2
        ;;
    Mar)
        Thang=3
        ;;
    Apr)
        Thang=4
        ;;
    May)
        Thang=5
        ;;
    Jun)
        Thang=6
```

```
        ;;
    Jul)
        Thang=7
        ;;
    Aug)
        Thang=8
        ;;
    Sep)
        Thang=9
        ;;
    Oct)
        Thang=10
        ;;
    Nov)
        Thang=11
        ;;
    *)
        Thang=12
        ;;
esac
echo "Hom nay la Thu $Thu, ngay $3, thang $Thang, nam $6"
echo "Thoi gian hien tai la $4"
```

Bài tập 5: Viết một script tính tổng 2 số nguyên đưa vào theo cách sau:

```
$tên_script số_nguyên_1 số_nguyên_2
```

Ví dụ script của bạn có tên *sum.sh* thì cách chạy và kết quả chạy như sau :

```
$./sum.sh 8 8
$16
```

Gợi ý nội dung tệp "*sum.sh*":

```
if [ $# -ne 2 ]
then
    echo "Chay file nhu sau - $0   x    y"
    echo "Trong do, x va y la hai so ma ban muon tinh
tong"
    exit 1
fi
    echo "Sum of $1 and $2 is $(($1 + $2))"
```