

10. Variables - Part II

Each time you run a script, the values of the variables are reset. This means that you can't rely on the values of variables from one script run to the next. This is why you should always initialize your variables at the start of your script.

For example, let's say you have a script called `var3.sh` that takes three arguments. The first argument is the name of the file, the second argument is the name of the directory, and the third argument is the name of the file extension. The script should print out the name of the file, the name of the directory, and the name of the file extension. Here's how you can write the script:

```
#!/bin/sh
echo "I was called with $# parameters"
echo "My name is $0"
echo "My first parameter is $1"
echo "My second parameter is $2"
echo "All parameters are $@"
```

Now, let's run the script with three arguments:

```
$ /home/steve/var3.sh
I was called with 0 parameters
My name is /home/steve/var3.sh
My first parameter is
My second parameter is
All parameters are
$
$ ./var3.sh hello world earth
I was called with 3 parameters
My name is ./var3.sh
My first parameter is hello
My second parameter is world
All parameters are hello world earth
```

As you can see, the script prints out the name of the file, the name of the directory, and the name of the file extension. This is because the script uses the `$0` variable to get the name of the script, the `$1` variable to get the first argument, the `$2` variable to get the second argument, and the `$@` variable to get all the arguments.

```
echo "My name is `basename $0`"
```

\$# \$1 ... \$2 . shift 9
 :

```
#!/bin/sh
while [ "$#" -gt "0" ]
do
    echo "$1 is $1"
    shift
done
```

□ □□□□ \$#□ 0□ □ □□ , □ □□ □□ □□ □□ shift□ □□□□ .

 \$?


```
#!/bin/sh
/usr/local/bin/my-command
if [ "$?" -ne "0" ]; then
    echo "Sorry, we had a problem there!"
fi
```

[illegible][illegible]

```

$ cd /tmp
$ cat my-script.$$.pid
PID(12345)
$ cat /tmp/my-script.$$
$ cat /tmp/my-script.$$.pid
PID(12345)
$ cat /tmp/my-script.$$.pid
PID(12345)

```

[illegible]

```
#!/bin/sh
old_IFS="$IFS"
IFS=:
```

```
echo "Please input some data separated by colons ..."
read x y z
IFS=$old_IFS
echo "x is $x y is $y z is $z"
```

❏ ❏❏❏❏ ❏❏ ❏ ❏❏❏❏ :

```
$ ./ifs.sh
Please input some data separated by colons ...
hello:how are you:today
x is hello y is how are you z is today
```

❏❏ ❏ "[hello:how are you:today:my:friend"❏❏ ❏❏❏ ❏❏ ❏ ❏❏❏❏ :

```
$ ./ifs.sh
Please input some data separated by colons ...
hello:how are you:today:my:friend
x is hello y is how are you z is today:my:friend
```

❏❏ IFS❏ ❏❏ ❏❏ ❏❏ , ❏❏ ❏ ❏❏ "❏❏❏ ❏ ❏❏ "❏❏❏ ❏❏❏ ❏ ❏❏❏ ❏❏ ❏❏❏❏ ❏❏
❏❏❏❏ . ❏❏❏ ❏❏❏❏ ❏❏ ❏❏ ❏❏❏❏ (❏❏ : old_IFS=\$IFS ❏❏ old_IFS="\$IFS").

Revision #1

Created 17 January 2024 04:15:01 by ❏❏❏❏ (MeatDumpling)

Updated 17 January 2024 04:15:14 by ❏❏❏❏ (MeatDumpling)