UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

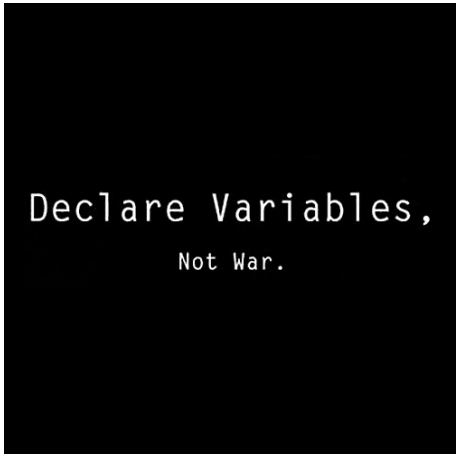
Utilizarea Sistemelor de Operare

31. Programare în shell

Programare în shell

- Variabile
- Instrucțiuni de decizie
- Instrucțiuni de ciclare
- Funcții

Variabile shell



- Inițializare obișnuită

```
a=10
b=ana
c="ana are mere"
d="$b $a"
```

- **NU** se folosește spațiu înainte și după egal la inițializare
  - s-ar considera comandă cu argumente
- Folosirea variabilelor se realizează prin prefixarea lor cu simbolul **$** (dollar)
- Atenție când se folosește $ (doar la utilizarea valorii)
- Variabilele shell **NU** au tip (întreg, șir, pointer)

Variabile speciale shell

- man bash
  - /Special Parameters
- $? – valoarea de retur a ultimei comenzi
- $$ - PID-ul procesului shell curent
- $! – PID-ul ultimului proces (job) lansat în background
- $_ - ultimul argument al ultimei comenzi

Variabile de mediu

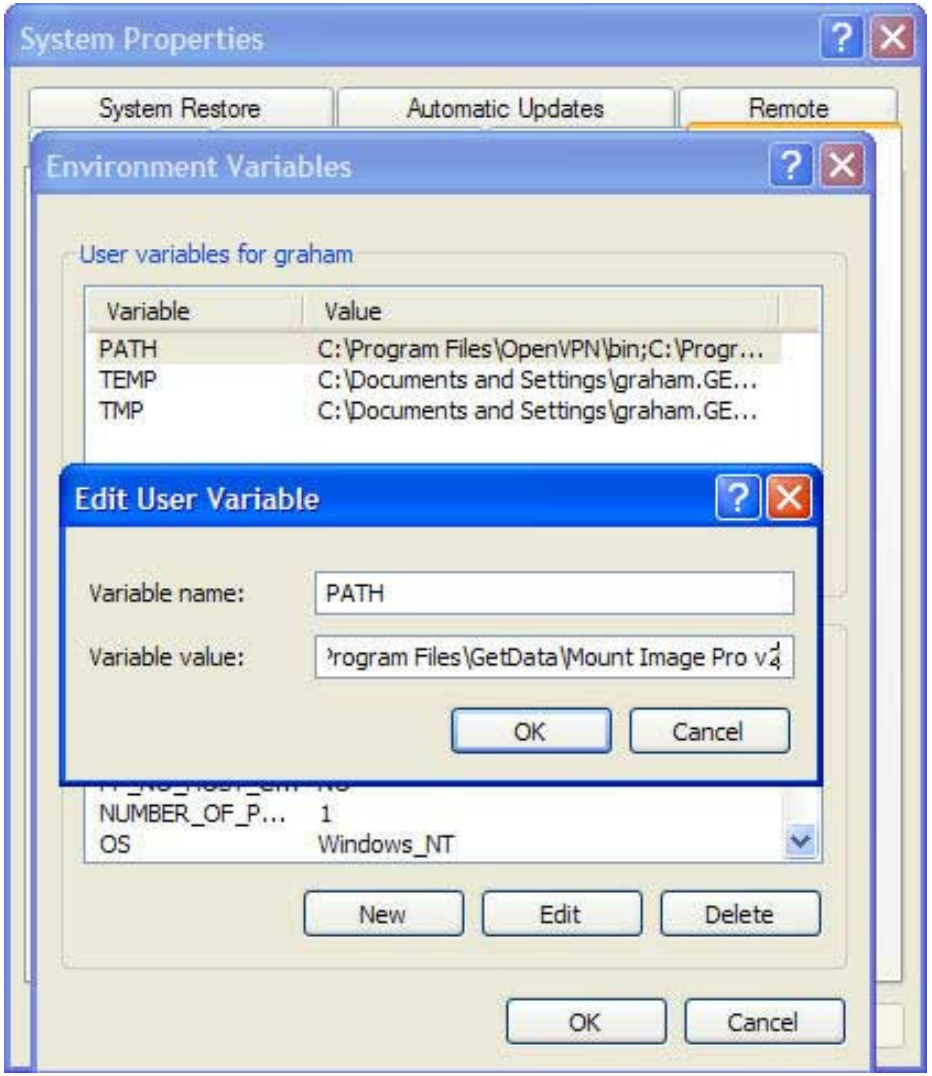
- Definesc contextul de rulare al unui proces
- Sunt moștenite de procesele copil
- Exemple: PATH, HOME, PWD, USERNAME, SHELL
- Operații cu variabile de mediu
  - listare: env, printenv
  - configurare variabilă ca variabilă de mediu (exportare)

```
$ export myvar
$ export myvar=42   # init and export
```

  - persistența configurării (startup)

Variabile de mediu în Windows

- per sistem: Control Panel -> System -> Advanced -> Environment Variables
- per utilizator: Control Panel -> Performance and Maintenance -> System



if

```
if grep 'dan' /etc/passwd; then
   echo "OK"
else
   echo "NOK"
fi

if test -f /path/to/file; then
   echo "File exists"
fi

if test $# -ne 1; then
   echo "Usage: $0 file"
fi
```

if (2)

- Primește ca argument o comandă
- 0 = succes -> condiția este îndeplinită
- !0 = insucces -> condiția nu este îndeplinită
- În general comanda test
  - `test -f file`
  - `test -d file`
  - `test string1 = string2`
  - `test num1 -eq num2`
  - `test -z string1`

for, seq

```
for i in 1 2 3 4 5 6 7 8 9 10; do … done

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

for i in $(seq 1 10); do … done

for i in $(seq -f "%02g" 1 10); do … done

for f in *; do … done

for user in $(cut -d ':' -f 1 < /etc/passwd); do …
  done

for arg in $@; do … done
```

while, read

```
while read a b c; do … done < file

IFS=':'; while read uname pass uid extra; do
  if test $uid -ge 1000; then
      echo "$uname"
  fi
done < /etc/passwd
```

while, read (2)

- Primește ca argument o comandă (la fel ca if)
  - de obicei read
- read
  - întoarce 0 când a citit o linie
  - întoarce diferit de zero la sfârșitul fișierului (EOF)
  - folosit
    - stand-alone (similar fgets, scanf)
    - împreună cu while
- IFS
  - input field separator
  - specifică argumentul care va fi folosit de read

Funcții în shell

```
hello()
{
    echo "hello"
}

ip_for_hostname()
{
    host "$1" | head -1 | cut -d ' ' -f 4
}

DEBUG()
{
    test "$_DEBUG" -eq 1 && $@
}
```

Funcții în shell (2)

- Modularizare
- func_name() { … }
- Se apelează ca o comandă (func_name)
- Poate folosi return
  - dacă nu, valoarea de retur a funcției este valoarea de retur a ultimei comenzi rulate
- Argumentele transmise sunt prelucrate cu ajutorul variabilelor specifice scripturilor shell
  - $#, $1, $2, $@