

Esercitazione 7

Gruppo LZ

Shell scripting

Agenda

Esempio

Esplorazione completa di una directory: script bash con ricorsione.

Esercizio 1

Esplorazione ricorsiva del file system

Esercizio 2

Esplorazione ricorsiva di più sottoalberi

Esempio – Script ricorsivi

Si scriva uno script bash avente interfaccia di invocazione

```
recurse_dir.sh dir
```

Il programma, dato un direttorio in ingresso **dir**, deve stampare su stdout l'elenco dei file contenuti nel direttorio e in tutti i suoi sottodirettori (analogamente al comando **ls -R**)

Schema di soluzione ricorsiva

`recurse_dir.sh arg1`

caso base

`arg1` è un file

→ ne stampo il nome assoluto

caso generale espresso in termini ricorsivi

`arg1` è una directory

→ mi muovo nella directory `arg1`;

per ogni file (normale o directory) **invoco nuovamente**
`recurse_dir.sh`

Bozza di soluzione

```
#!/bin/bash
```

```
if ! test -d "$1" ; then  
    echo `pwd` /$1
```

Caso
base

```
else  
    cd "$1"  
    for f in * ; do  
        "$0" "$f"  
    done  
fi
```

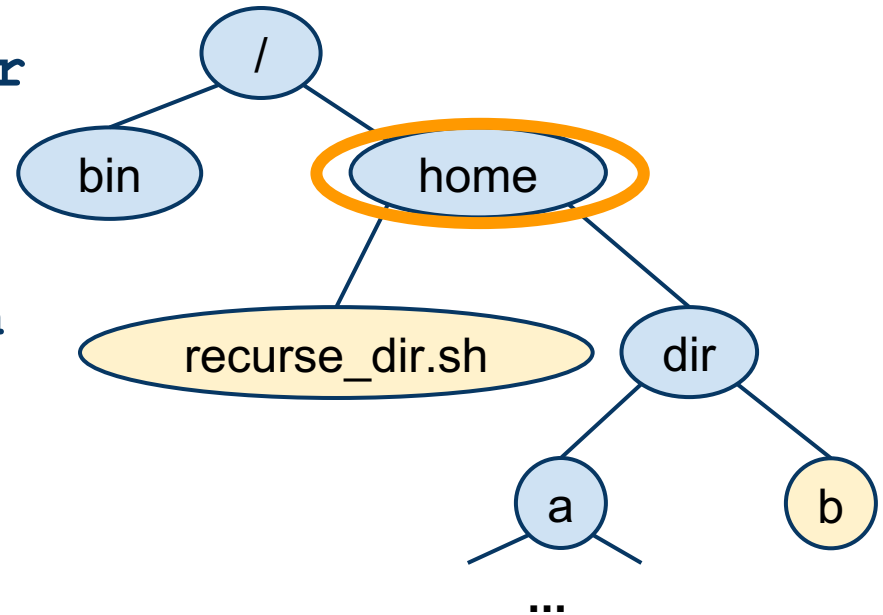
Caso
generale

Chiamata ricorsiva

Ricorsione (1/6)

```
$ pwd  
/home  
$ /home/recurse_dir.sh dir
```

```
➔ if ! test -d "$1" ; then  
    echo `pwd` /$1  
else  
    cd "$1"  
    for f in * ; do  
        "$0" "$f"  
    done  
fi
```



□ directory
□ file

VARIABILI:

\$PWD /home

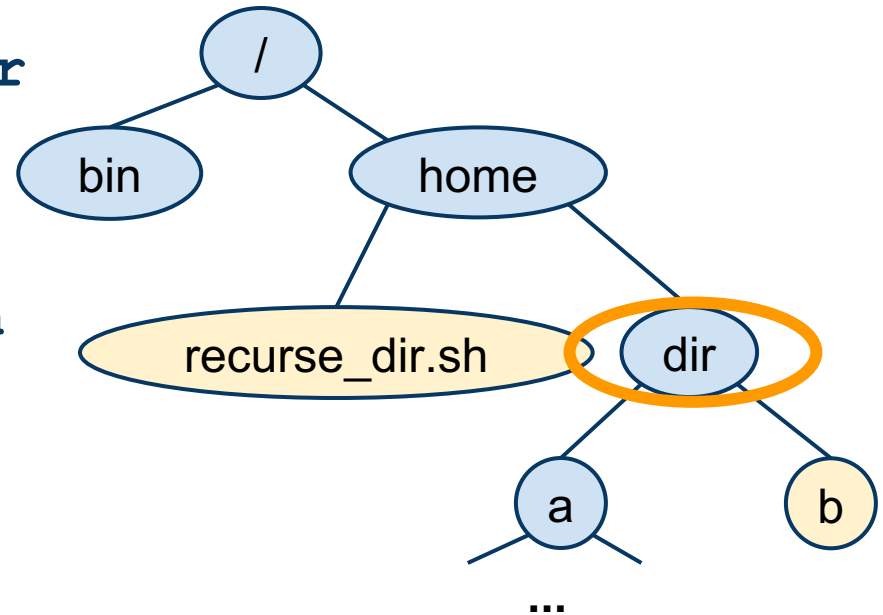
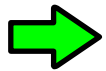
\$0 /home/recurse_dir.sh

\$1 dir

Ricorsione (2/6)

```
$ pwd
/home
$ /home/recurse_dir.sh dir
```

```
if ! test -d "$1" ; then
    echo `pwd` /$1
else
    cd "$1"
    for f in * ; do
        "$0" "$f"
    done
fi
```



□ directory
□ file

VARIABILI:

\$PWD /home/dir

\$0 /home/recurse_dir.sh

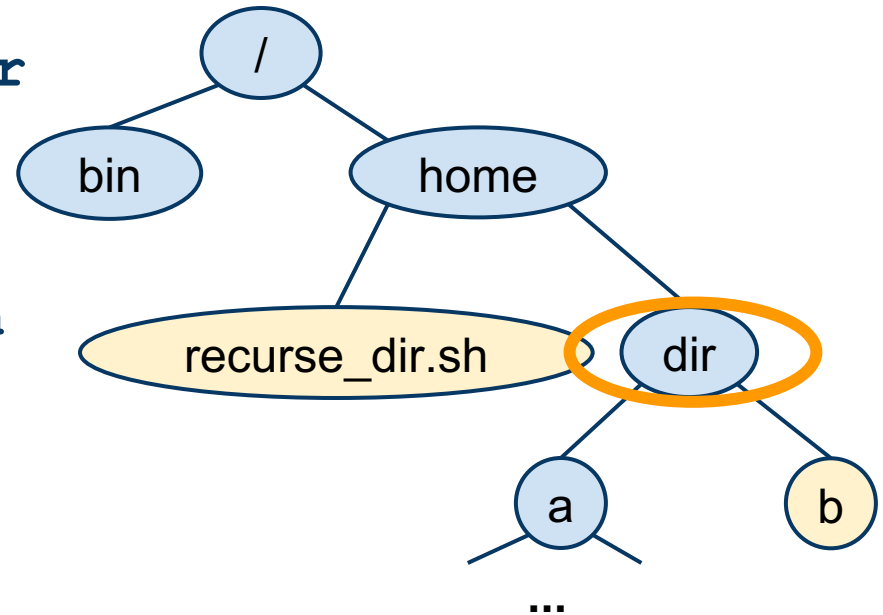
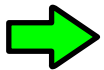
\$1 dir

Ricorsione (3/6)

```
$ pwd
/home
$ /home/recurse_dir.sh dir
```

```
$ /home/recurse_dir.sh a
```

```
if ! test -d "$1" ; then
    echo `pwd` /$1
else
    cd "$1"
    for f in * ; do
        "$0" "$f"
    done
fi
```



□ directory
□ file

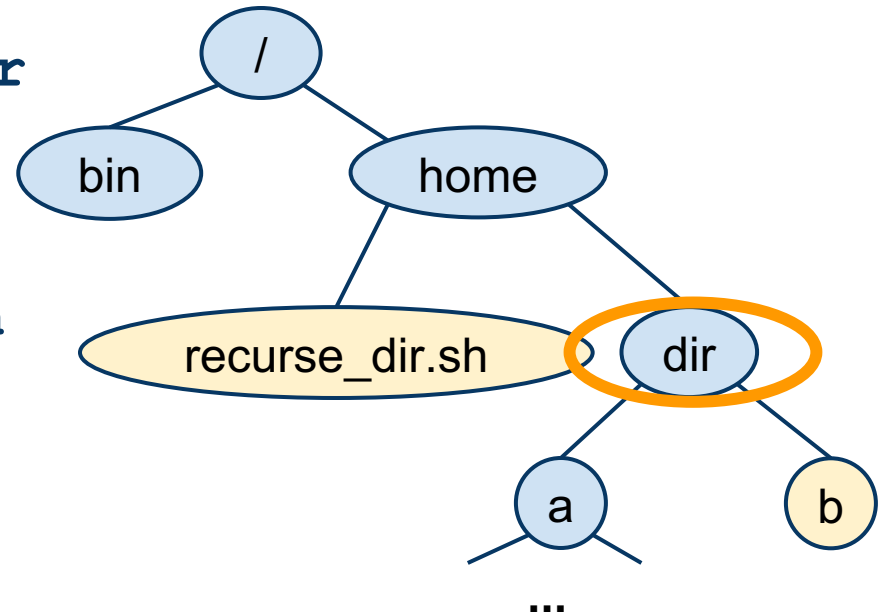
VARIABILI:

\$PWD /home/dir
\$0 /home/recurse_dir.sh
\$1 dir

Ricorsione (4/6)

```
$ pwd
/home
$ /home/recurse_dir.sh dir
```

```
➔ if ! test -d "$1" ; then
    echo `pwd` /$1
else
    cd "$1"
    for f in * ; do
        "$0" "$f"
    done
fi
```



□ directory
□ file

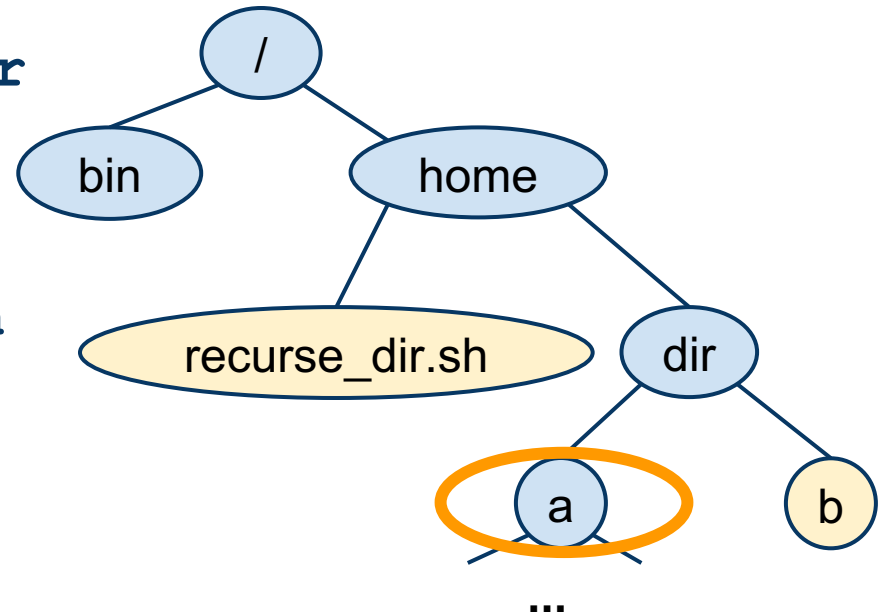
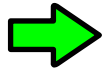
VARIABILI:

```
$PWD /home/dir
$0 /home/recurse_dir.sh
$1 a
```

Ricorsione (5/6)

```
$ pwd
/home
$ /home/recurse_dir.sh dir
```

```
if ! test -d "$1" ; then
    echo `pwd` /$1
else
    cd "$1"
    for f in * ; do
        "$0" "$f"
    done
fi
```



□ directory
□ file

VARIABILI:

\$PWD /home/dir/a

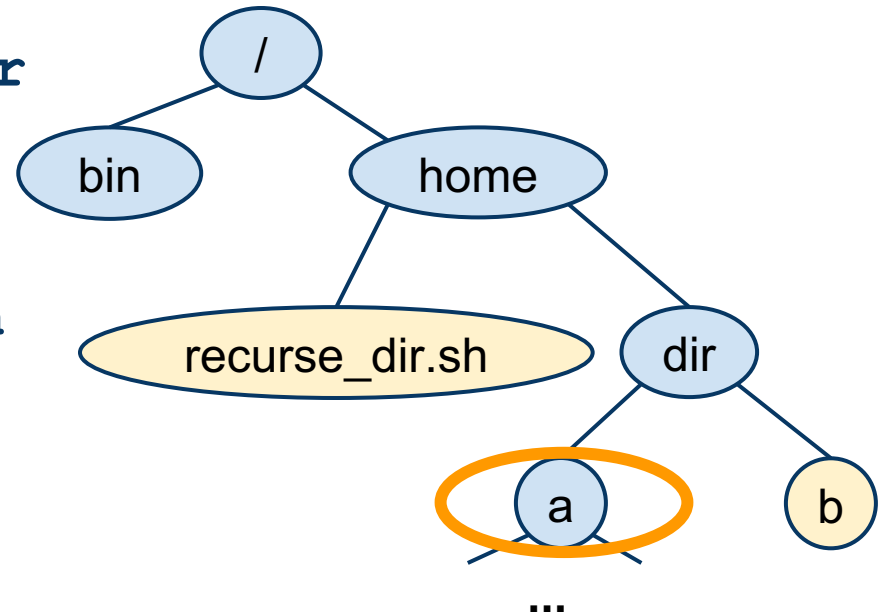
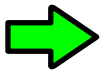
\$0 /home/recurse_dir.sh

\$1 a

Ricorsione (6/6)

```
$ pwd
/home
$ /home/recurse_dir.sh dir
```

```
if ! test -d "$1" ; then
    echo `pwd` /$1
else
    cd "$1"
    for f in * ; do
        "$0" "$f"
    done
fi
```



□ directory
□ file

VARIABILI:

\$PWD /home/dir/a

\$0 /home/recurse_dir.sh

\$1 ...

ATTENZIONE

Nell'esempio lo script è stato invocato **specificando il suo nome assoluto**:

```
$ /home/recurse_dir.sh dir
```

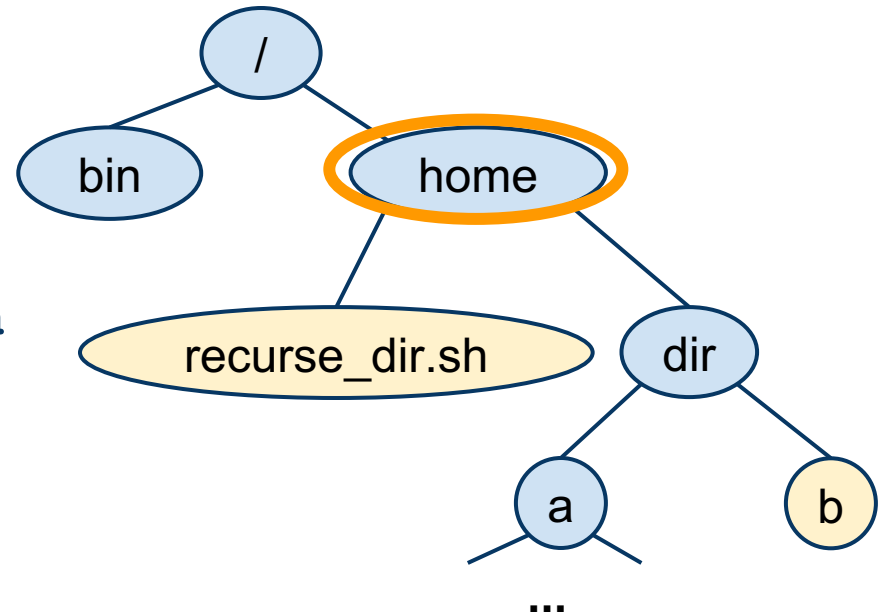
Cosa succederebbe invocandolo
con un **nome relativo**?

```
$ ./recurse_dir.sh dir
```

Ricorsione - alternativa (1/3)

```
$ pwd  
/home  
$ ./recurse_dir.sh dir
```

```
if ! test -d "$1" ; then  
    echo `pwd` /$1  
else  
    cd "$1"  
    for f in * ; do  
        "$0" "$f"  
    done  
fi
```



□ directory
□ file

VARIABILI:

\$PWD /home

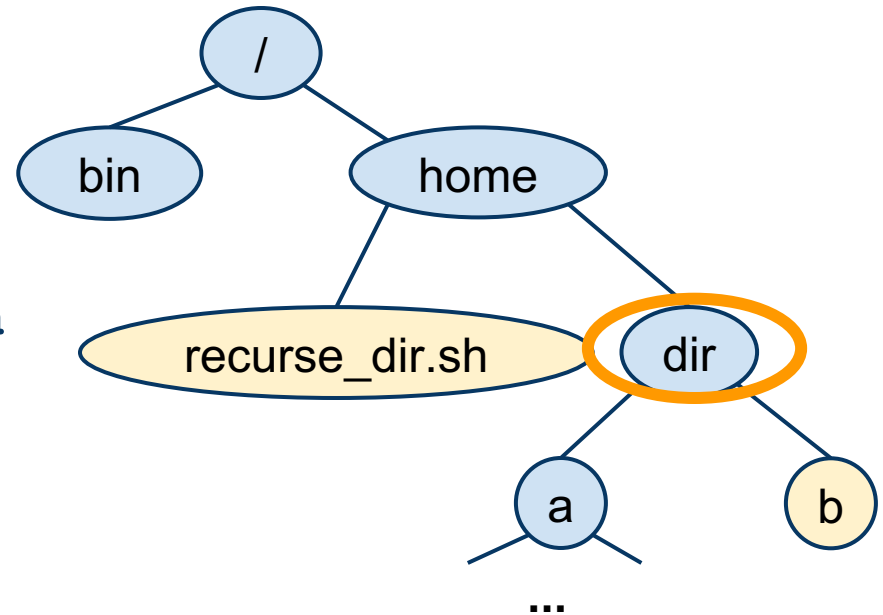
\$0 ./recurse_dir.sh

\$1 dir

Ricorsione - alternativa (2/3)

```
$ pwd  
/home  
$ ./recurse_dir.sh dir
```

```
if ! test -d "$1" ; then  
    echo `pwd` /$1  
else  
    ➔ cd "$1"  
    for f in * ; do  
        "$0" "$f"  
    done  
fi
```



□ directory
□ file

VARIABILI:

\$PWD /home/dir

\$0 ./recurse_dir.sh

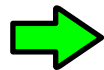
\$1 dir

Ricorsione - alternativa (3/3)

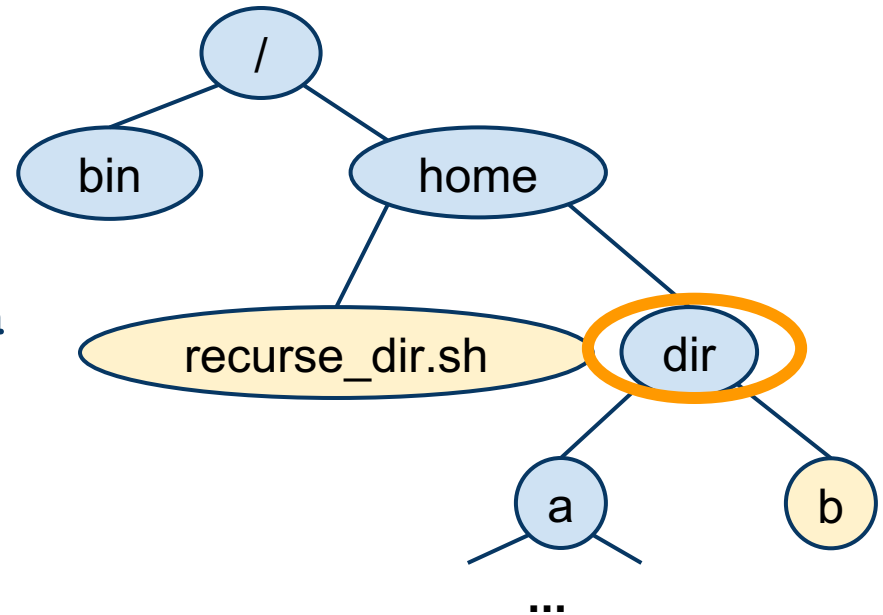
```
$ pwd
/home
$ ./recurse_dir.sh dir
```

```
$ ./recurse_dir.sh a
```

```
if ! test -d "$1" ; then
    echo `pwd` /$1
else
    cd "$1"
    for f in * ; do
        "$0" "$f"
    done
fi
```



non
funziona!



□ directory
□ file

VARIABILI:

```
$PWD /home/dir
$0 ./recurse_dir.sh
$1 dir
```

Come risolvere?

Problema: Un valore dipendente dalla directory di lavoro corrente (un percorso relativo) viene "**propagato**" da una invocazione ricorsiva all'altra (tramite la variabile \$0)

La directory di lavoro però cambia (perchè usiamo il comando `cd` nel codice)

Possibile soluzione: Prima di iniziare la ricorsione memorizzare la directory di partenza in una variabile che verrà usata per le invocazioni ricorsive

Occorre creare:

- **Script ricorsivo**
- **Script di invocazione:**
 - Controlla i parametri
 - Salva in maniera "stabile" il percorso dello script ricorsivo
 - Innesca la ricorsione

Struttura di un file comandi ricorsivo

invoker.sh

#!/bin/sh

Controllo degli argomenti

Invocazione del file comandi ricorsivo do_recursive.sh

do_recursive.sh

#!/bin/sh

Esecuzione del compito

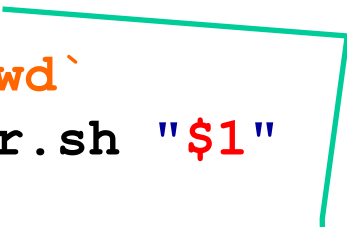
Invocazione del file comandi ricorsivo do_recursive.sh

Script di invocazione

recurse_dir.sh

```
#!/bin/bash
# ... controllo argomenti

oldpath=$PATH
PATH=$PATH:`pwd`
do_recurse_dir.sh "$1"
PATH=$oldpath
```



do_recurse_dir.sh

```
#!/bin/bash
if ! test -d "$1" ; then
    echo `pwd`/$1
else
    cd "$1"
    for f in * ; do
        "$0" "$f"
    done
fi
```

PATH è una variabile d'ambiente che contiene dei nomi di directory separati da ":".

Quando lancio un comando senza alcun path (né assoluto né relativo, es: invoco **ls** invece di **/bin/ls**), il SO cerca quel comando in tutte le directory contenute nella variabile **PATH**.

Script di invocazione

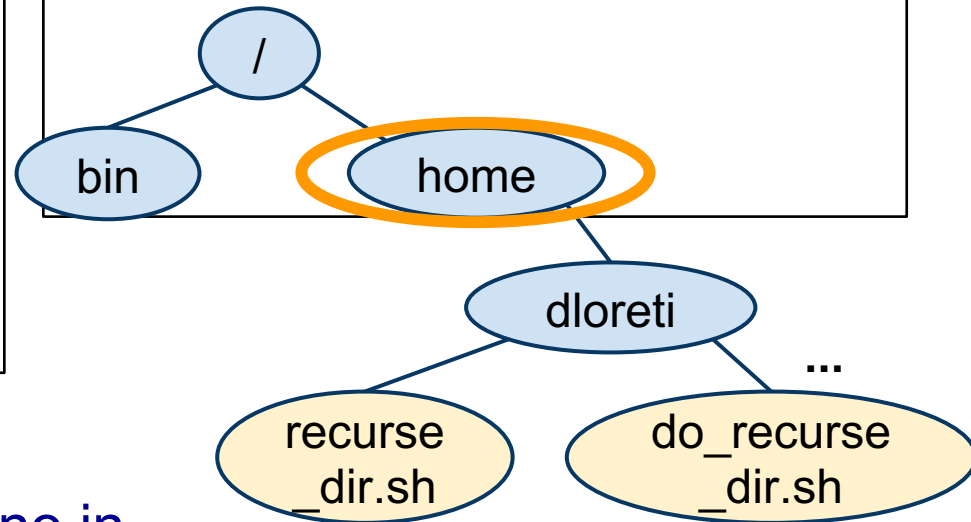
recurse_dir.sh

```
#!/bin/bash
# ... controllo argomenti

oldpath=$PATH
PATH=$PATH:`pwd`
do_recurse_dir.sh "$1"
PATH=$oldpath
```

do_recurse_dir.sh

```
#!/bin/bash
if ...
```



Problema :

Che succede se gli script si trovano in
/home/dloreti e l'utente li invoca dalla directory corrente
/home con il path relativo: ./dloreti/recurse_dir.sh

=> `pwd` viene espanso in "/home"

=> **PATH=\$PATH:/home**

=> do_recurse_dir.sh viene cercato in /home

→ **NON TROVATO!**

Soluzione generale

recurse_dir.sh

```
#!/bin/bash
# ... controllo argomenti

if [[ "$0" = /* ]] ; then
    # se $0 è un path assoluto
    dir_name=`dirname "$0"`
    recursive_cmd="$dir_name/do_recurse_dir.sh"
elif [[ "$0" = */* ]] ; then
    # se c'è uno slash, ma non inizia con /
    # $0 è un path relativo
    dir_name=`dirname "$0"`
    recursive_cmd "`pwd`/$dir_name/do_recurse_dir.sh"
else
    # Non si tratta nè di un path relativo, nè di uno
    # assoluto, il comando $0 sarà cercato in $PATH.
    recursive_cmd=do_recurse_dir.sh
fi
#Invoco il comando ricorsivo
"$recursive_cmd" "$1"
```

Restituisce \$0 tranne
l'ultimo / e ciò che segue

do_recurse_dir.sh

```
#!/bin/bash
if ...
```

Soluzione generale

recurse_dir.sh

```
#!/bin/bash
# ... controllo argomenti
```

```
if [[ "$0" = /* ]] ; then
    # se $0 è un path assoluto
    dir_name=`dirname "$0"`
    recursive_cmd="$dir_name/do_recurse_dir.sh"
```

```
elif [[ "$0" = */* ]] ; then
```

```
    # se c'è un
```

```
    # $0 è un p
```

```
    dir_name=`d
```

```
    recursive_c
```

```
else
```

```
    # Non si tratta ne di un path relativo, ne di uno
```

```
    # assoluto, il comando $0 sarà cercato in $PATH.
```

```
    recursive_cmd=do_recurse_dir.sh
```

```
fi
```

```
#Invoco il comando ricorsivo
```

```
"$recursive_cmd" "$1"
```

Esempio:

\$0=/home/recurse_dir.sh

`dirname "\$0"` → /home

\$recursive_cmd=/home/do_recurse_dir.sh

Soluzione generale

recurse_dir.sh

```
#!/bin/bash
# ... controllo argomenti

if [[ "$0" = /* ]] ; then
    # se $0 è un path assoluto
    dir_name=`dirname "$0"`
    recursive_cmd="$dir_name/do_recurse_dir.sh"
elif [[ "$0" = */* ]] ; then
    # se c'è uno slash, ma non inizia con /
    # $0 è un path relativo
    dir_name=`dirname "$0"`
    recursive_cmd="`pwd`/$dir_name/do_recurse_dir.sh"
else
    # Non
    # asso
    recurs
fi
#Invoco
"$recursive_cmd" "$1"
```

Esempio:

\$0=../folder/recurse_dir.sh

`dirname "\$0"` → ../folder

\$recursive_cmd=/home/../folder/do_recurse_dir.sh

Soluzione generale

recurse_dir.sh

```
#!/bin/bash
# ... controllo argomenti

if [[ "$0" = /* ]] ; then
    # se $0 è un path assoluto
    dir_name=`dirname "$0"`
    recursive_cmd="`pwd`/$dir_name/do_recurse_dir.sh"
```

Esempio:

\$0=recurse_dir.sh

\$recursive_cmd=do_recurse_dir.sh

```
    dir_name=`dirname "$0"`
    recursive_cmd="`pwd`/$dir_name/do_recurse_dir.sh"
```

else

```
# Non si tratta nè di un path relativo, nè di uno
# assoluto, il comando $0 sarà cercato in $PATH.
```

```
recursive_cmd=do_recurse_dir.sh
```

fi

```
#Invoco il comando ricorsivo
```

```
"$recursive_cmd" "$1"
```

Esercizio 1 (1/2)

Realizzare un file comandi (ricorsivo) che abbia la sintassi

`contaOcc dir inStr str`

dove:

- `dir` è il nome assoluto di un direttorio esistente nel file system
- `inStr` e `str` sono due stringhe

Esercizio 1 – (2/2)

Il compito del file comandi è quello di :

- Esplorare ricorsivamente il sottoalbero individuato da **dir**
- Individuare tutti i file ordinari il cui nome contenga la stringa **inStr**;
- Per ogni file che rispetta tale caratteristica contare il numero **X** di occorrenze della stringa **str** e scrivere nel file *report.log* (situato nella home dell'utente che ha invocato lo script) una stringa del tipo:

Il file <absNameFile> contiene <X> occorrenze di <str>

Dove:

- *<absNameFile> è il nome assoluto del file*
- *<X> è il numero occorrenze individuate in <absNameFile>*
- Infine, per ogni directory nel sottoalbero individuato da **dir** (inclusa **dir** stessa) stampare il numero di file il cui nome contenga la stringa **inStr**

Esercizio 1: Suggerimenti (1/2)

Prima di tutto realizzare la ricorsione e testare che funzioni correttamente!

Poi:

- Contare il numero di occorrenze di una stringa in un file: si veda l'opzione `-o` del comando **grep**
 - ▣ Verificare il comportamento da linea di comando. Come contare le occorrenze?
- `<absNameFile>` deve riportare il path assoluto del file.
 - ▣ Lo script userà `cd` per saltare in ogni direttorio (e poi analizzare ogni file in esso contenuto)
 - ▣ Ad ogni iterazione l'absolute name del file può essere ricostruito sulla base del direttorio corrente

Esercizio 1: Suggerimenti (2/2)

- L'output dello script dovrà essere fatto aggiungendo righe a *report.log*
 - Cosa succede se il file esiste già al momento dell'invocazione? Qualora esista, assicurarsi anche che sia vuoto all'inizio dell'esecuzione.
- Per ogni directory in **dir** (inclusa) contare il numero di file il cui nome contiene la stringa **inStr**
 - In quale file deve essere messa la stampa? Nell'invoker o nel file che attua la ricorsione?

Esercizio 2 (1/2)

Realizzare un file comandi (ricorsivo) che abbia la sintassi

contaParole Fout dir1...dirN

dove:

- **Fout** è il nome assoluto di un file esistente o non esistente nel file system
- **dir1...dirN** sono nomi assoluti di direttori esistenti nel file system

Il compito del file comandi è quello di eseguire una ricerca in tutte le directory **dir1...dirN**. Per ogni directory **dir_i** :

- Scandire il contenuto della directory e, per ogni file ordinario trovato, contare il numero **N** di parole in esso contenute.

Esercizio 2 – (2/2)

- Qualora **N** sia pari, lo script deve scrivere in **Fout** una stringa del tipo:
Report <pid>: Il file <absNameFile> contiene <N> parole
Dove:
 - ▣ **<pid>** è il PID del processo che esegue lo script
 - ▣ **<absNameFile>** è il nome assoluto del file
 - ▣ **<N>** è il numero (pari) di parole individuate in **<absNameFile>**
- Qualora **Fout** esista già, il suo contenuto NON deve essere sovrascritto, ma le line aggiunte in fondo al file
- Al termine della ricerca su tutte le directory, il file comandi dovrà **stampare su stdout il numero totale di file** che contengono un numero **N** pari di parole **trovati nelle directory esplorate**

Esercizio 2: Suggerimenti (1/2)

- Invoker deve iterativamente attivare lo script ricorsivo sulle directory date. Ricordare:
 - "\$@" (o \$*) → lista delle variabili posizionali
 - shift → scorrimento a sinistra delle var posizionali
- Contare il numero di parole in un file:
 - Verificare qual è l'output di `wc` prima di inserirlo nello script:

```
utente~$ wc -w myfile  
53      myfile
```
 - Come ovviare a questo problema?

Esercizio 2: Suggerimenti (2/2)

- Il pid del processo che esegue uno script è reperibile tramite un'opportuna variabile notevole. Ricordare:
 - un processo esegue l'invoker (**contaParole**), altri processi eseguono lo script ricorsivo.
 - La stampa su **Fout** deve essere posizionata nello script ricorsivo, ma deve includere il pid dell'invoker. Come fare?
- Ricavare il numero totale di file contenenti un numero pari di parole tramite opportuno filtraggio e conteggio delle linee di **Fout**