

Settima esercitazione

Shell scripting

Agenda

Esempio 1

Creazione di un semplice script bash per l'esplorazione del file system

Esempio 2

Script bash con ricorsione: esempio guidato

Esercizio 1 - DA SVOLGERE

Esplorazione ricorsiva del file system

Esercizio 2 - DA SVOLGERE

Esempio di compito d'esame

Esempio 1- Script bash per l'esplorazione del file system

Creare un file comandi Unix con la seguente interfaccia

summary.sh dir

Il file comandi dovrà scandire il contenuto del directory **dir** e dovrà stamparne un sommario del contenuto sul file **summary.out**

In particolare, **per ciascun elemento trovato in dir**

- nel caso in cui si tratti di un "**regular file**": riportare il nome del file ed i primi 10 caratteri (byte)
- se è una **directory**: riportare il nome del directory ed il numero di direttori/file contenuti

Note alla soluzione

E' necessario **iterare** su tutti gli elementi di un direttorio

- Ciclo **for** opportuno

E' necessario gestire **due distinte condizioni:**

- Caso **file**

`head -c 10 nomefile` → Primi 10 caratteri

- Caso **directory**

`ls -l nomedir` → stampa a video il contenuto della directory **nomedir** (un elemento per riga)

`wc -l` → conta le righe (da file o stdin)

`ls -l nomedir | wc -l`

→ conta gli elementi contenuti in nomedir

Soluzione

```
#!/bin/bash
if test $# -ne 1 ; then
    echo "Usage: $0 dir"
    exit
fi
if ! test -d "$1" ; then
    echo "$1 is not a valid directory"
    exit
fi
cd "$1"
for i in * ; do
    if test -d "$i" ; then
        echo $i: `ls -l "$i" | wc -l` elementi >> summary.out
    elif test -f "$i" ; then
        echo $i: `head -c 10 "$i"` >> summary.out
    fi
done
```

Soluzione

```
#!/bin/bash
if test $# -ne 1 ; then
    echo "Usage: $0 dir"
    exit
fi
if ! test -d "$1"; then
    echo "$1 is not a valid directory"
    exit
fi
cd "$1"
for i in * ; do
    if test -d "$i"; then
        echo $i: `ls -l "$i" | wc -l` elementi >> summary.out
    elif test -f "$i" ; then
        echo $i: `head -c 10 "$i"` >> summary.out
    fi
done
```

Perchè non chiude la shell?!

Perchè mi conviene usare i doppi apici?

Soluzione

```
#!/bin/bash
if test $# -ne 1 ; then
    echo "Usage: $0 dir"
    exit
fi
if ! test -d "$1" ; then
    echo "$1 is not a valid directory"
    exit
fi
cd "$1"
for i in * ; do
    if test -d "$i" ; then
        echo "$i": `ls -l "$i" | wc -l` elementi >> summary.out
    elif test -f "$i" ; then
        echo "$i": `head -c 10 "$i"` >> summary.out
    fi
done
```

E se invece scrivessi:

```
# cd "$1"
for i in "$1"/* ; do
```

Dove verrebbe messo summary.out ?

Un'estensione possibile

Creare un file comandi Unix con la seguente interfaccia

summary.sh dir filter

Il file comandi dovrà

- operare la stessa logica dell'esercizio precedente
- escludere (dalla scrittura su **summary.out**) directory o file che **inizino** per la stringa **filter**

Soluzione

```
#!/bin/bash
cd "$1"
for i in * ; do
    case "$i" in
        $2*)
            ;;
        *)
            if test -d "$i" ; then
                echo "$i": `ls "$i" | wc -l` elementi >> summary.out
            elif test -f "$i" ; then
                echo "$i": `head -c 10 "$i"` >> summary.out
            fi
            ;;
    esac
done
```

Digressione: alternative a test

- `if test $# -ne 2; then`

commando `test`: invocato in un'altra shell (fork+exec). Il valore di ritorno viene sostituito all'espressione di `test` per discriminare l'ingresso nell'`if`.

- `if [$# -ne 2] ; then`

commando `[`: EQUIVALENTE a `test`.

- `if [[$# != 2]] ; then`

commando `[[`: la condizione è valutata nella shell stessa (nessuna fork!). Vantaggi:

- Posso usare i metacaratteri: `if [[$1 = f*]] ; then`
- Posso usare `>` `<` per confrontare stringhe

`>` `<` eseguono un confronto lessicografico fra stringhe.

Per confrontare numeri devo usare sempre `-lt` `-gt` `-le` `-ge`

- Posso usare operatori logici:

```
if [[ $a == $b && $a != $c ]] ; then  
if [[ $a == $b || $a != $c ]] ; then
```

Esempio 2 - Script ricorsivi

Si scriva uno script bash avente interfaccia di invocazione

reurse_dir.sh dir

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

Schema di soluzione ricorsiva

`recurse_dir.sh arg1`

caso **base**

arg1 è un file

→ stampo il nome

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
```

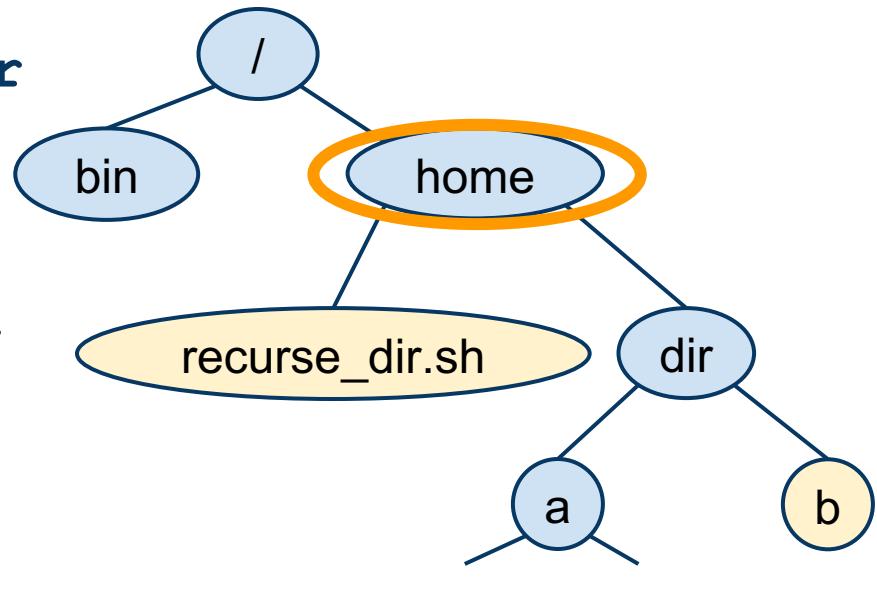
Caso
generale

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

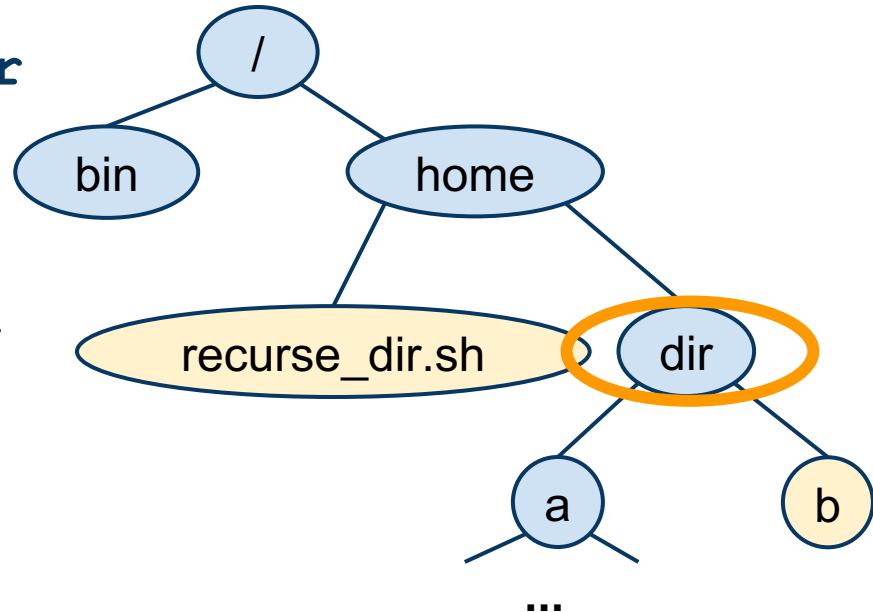
\$0 /home/recurse_dir.sh

\$1 dir

Ricorsione (2/6)

```
$ pwd  
/home  
$ /home/recuse_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/recuse_dir.sh

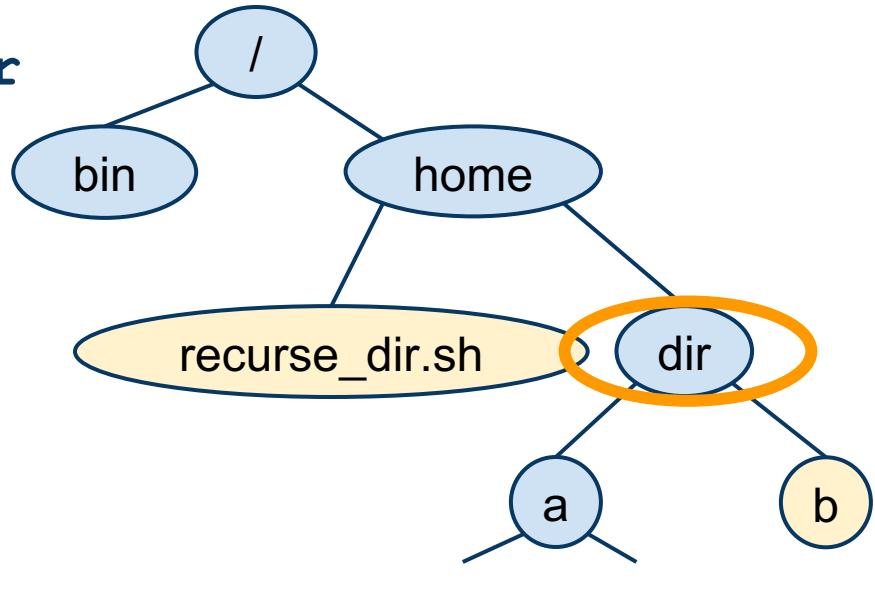
\$1 dir

Ricorsione (3/6)

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

```
/home/recuse_dir.sh a
```

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



■ directory
■ file

VARIABILI:

\$PWD /home/dir

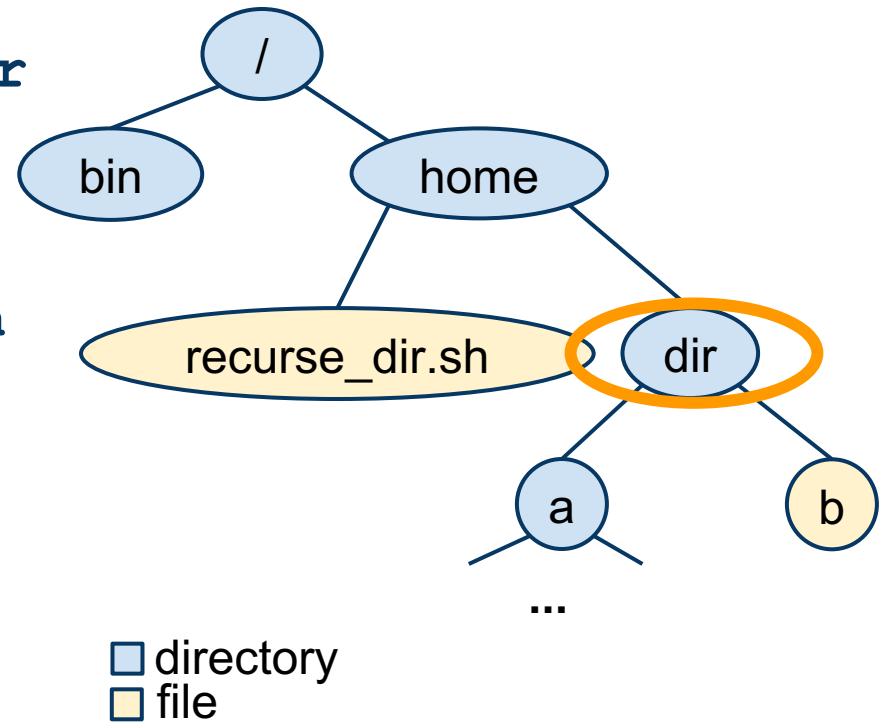
\$0 /home/recuse_dir.sh

\$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
```



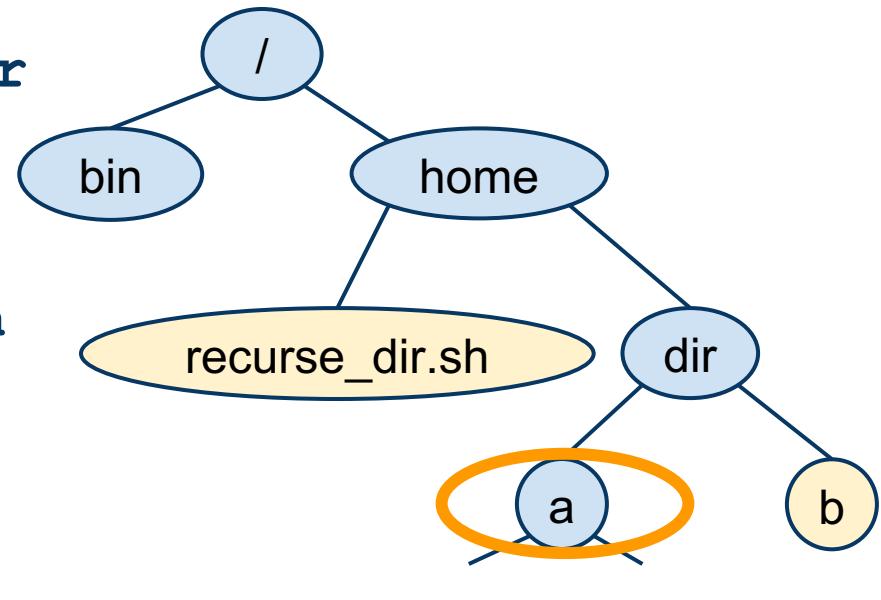
VARIABILI:

\$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
```



VARIABILI:

\$PWD /home/dir/a

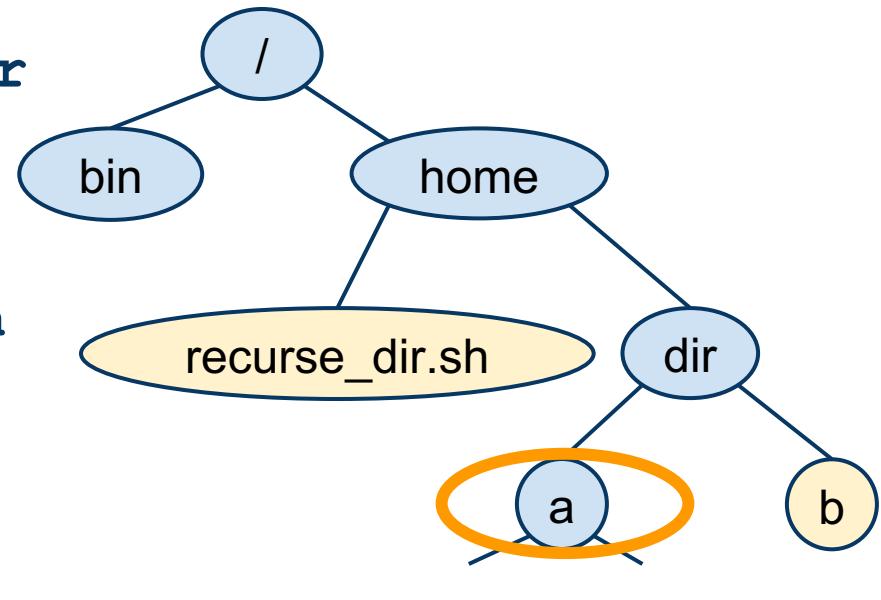
\$0 /home/recurse_dir.sh

\$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 path assoluto:

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

Cosa succederebbe invocandolo
con un path relativo?

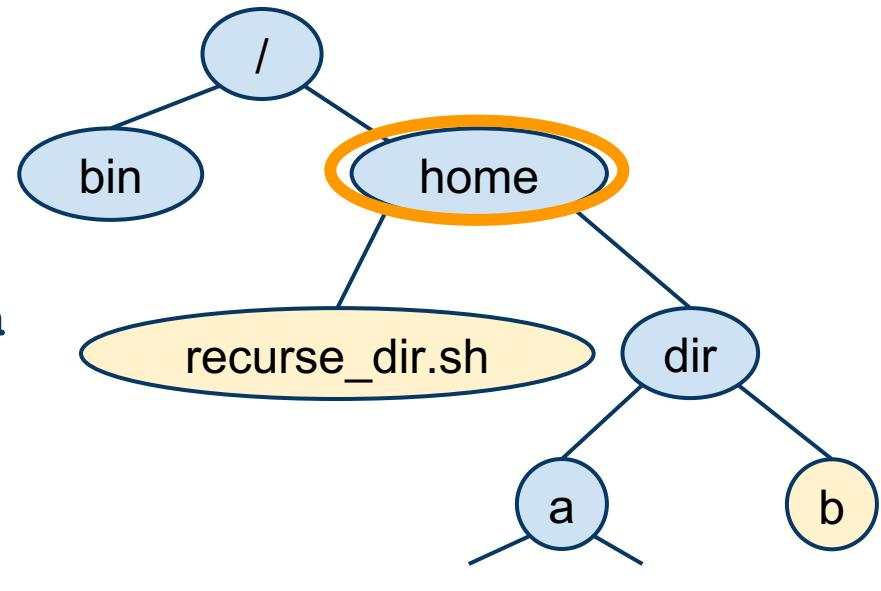
```
~$ ./recurse_dir.sh dir
```



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

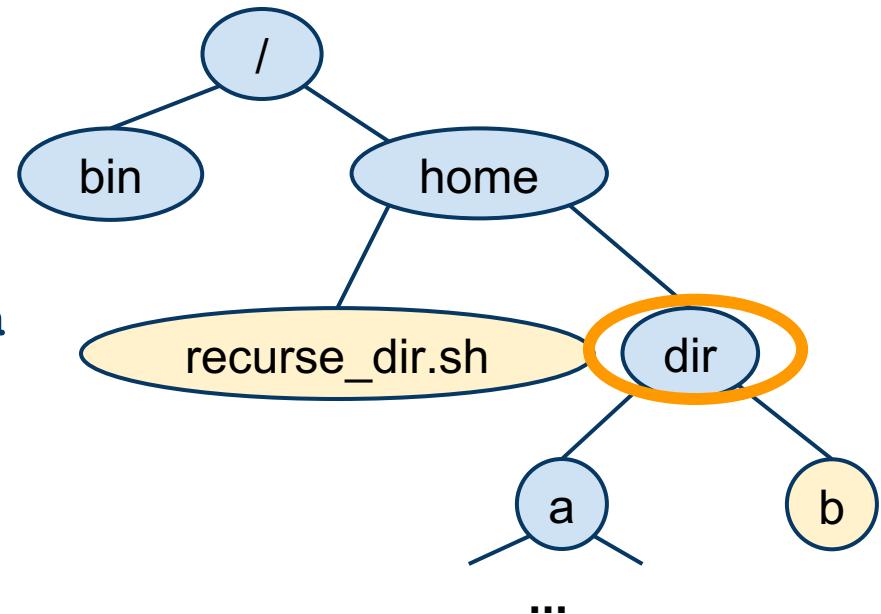
VARIABILI:

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

Ricorsione - alternativa (2/3)

```
$ pwd  
/home  
$ ./re 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:

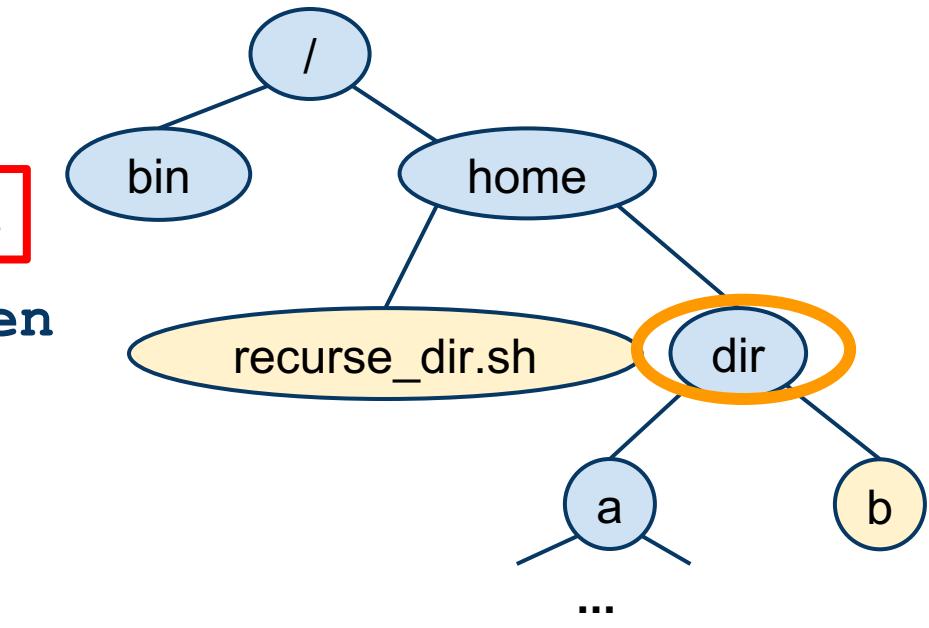
\$PWD /home/dir
\$0 ./re recurse_dir.sh
\$1 dir

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
```



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

re 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 path di directory separate da ":".

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

Script di invocazione

`recurse_dir.sh`

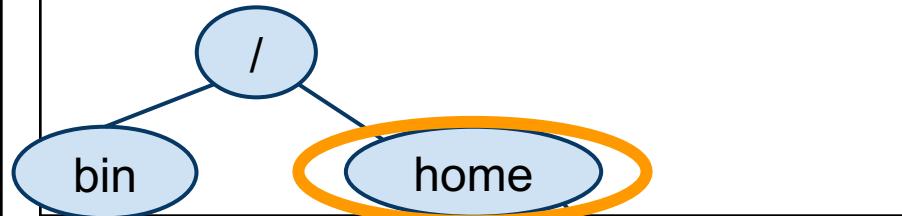
```
#!/bin/bash
# ... controllo argomenti

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/anna` e l'utente li invoca dalla directory corrente
`/home` con il path relativo: `./anna/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 né di un path relativo, né 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 ...
```

Esercizio 1 – Esplorazione ricorsiva del file system (1/2)

Realizzare un file comandi (ricorsivo) che abbia la sintassi

search.sh minR C dir1 dir2 ... dirN

Elenco (di lunghezza non nota a priori) di direttori

dove:

dir1 ... dirN sono un numero N qualsiasi, non noto a priori, di **nomi di direttori assoluti** che devono esistere nel file system.

minR è un intero

C è un singolo carattere

Esercizio 1 - (2/2)

Il compito del file comandi è quello di :

- Visitare (ricorsivamente) tutti i **sottoalberi** individuati da **dir1 ... dirN**
- Per ogni file ordinario trovato verificare che:
 - il **numero di occorrenze del carattere C** sia **maggior o uguale a MinR**
 - Il **proprietario** del file sia **l'utente che esegue il comando**
- Scrivere il nome assoluto di ciascun file che soddisfi il precedente requisito in un **file di output nella home** dell'utente che ha invocato il comando

Suggerimenti es.1 (1/2)

Devo ciclare sui parametri **dir1 ... dirN** in ingresso. Mi occorrono:

- ⇒ una **variabile** che contenga **l'elenco dei parametri in ingresso** (insieme di tutte le variabili posizionali)
- ⇒ un comando per **scartare i primi tre parametri. Quale?**

NB: le **soluzioni** degli esempi e dell'esercizio contengono numerosi suggerimenti utili sotto forma di commento. Vale la pena darci un'occhiata anche se la vostra soluzione funziona!

Suggerimenti es.1 (2/2)

Devo contare il **numero di occorrenze del carattere C** in ogni file ordinario trovato

grep -c <string> → conta il numero di righe che hanno almeno un'occorrenza di **<string>** => non va bene!

Occorre una opzione di grep che mi permetta di filtrare **solo le stringhe che fanno match** con la stringa passata (nel nostro caso il carattere C).

Poi **conterò** il numero di elementi emessi in output da grep.

Suggerimenti es.1 (3/2)

Qual è lo username dell'utente che esegue?
(v. variabili di ambiente...)

Come ricavare lo username del proprietario di un file?

Varie soluzioni:

- Uso del comando **stat** per ottenere lo username del proprietario del file (**man stat**, v. opzione **--format**)
- Filtraggio dell'output di ls tramite il comando **cut**
- Filtraggio con il comando **awk**

Esercizio 2

Si realizzi un file comandi bash con la seguente interfaccia:

seleziona dir start N

dove:

- **dir** è il path di una directory esistente nel file system
- **start** è una stringa di caratteri
- **N** è un intero positivo

Dopo aver effettuato gli opportuni controlli sui parametri di ingresso, il file comandi deve analizzare ricorsivamente il sotto-albero individuato dalla directory **dir**.

In particolare, per ogni sotto-directory di **dir**, deve contare il numero X di file il cui nome inizia per **start** e selezionare solo le sotto-directory per cui $X > N$.

Lo script deve scrivere nel file **esito.out** i nomi assoluti delle sole sotto-directory selezionate (ovvero solo quelli contenenti più di N file che iniziano per **start**) ognuno con il relativo valore di X nel seguente formato:

<nome assoluto directory> X

Il file **esito.out** dovrà essere salvato nel direttorio da cui il programma **seleziona** è stato chiamato.