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 3 – DA SVOLGERE**
  - Esplorazione ricorsiva del file system

Due parole "pratiche" su shell Unix

- Doppia natura
  - **Interprete di Comandi**: permette di eseguire programmi di sistema e programmi utente
  - **Linguaggio di programmazione**: permette ai comandi di essere combinati per formare **nuovi comandi**

- Esecuzione **Interattiva**
  - Legge le istruzioni inserite da standard input

- Esecuzione **Non-Interattiva**
  - Legge le istruzioni da un file di comandi: **script di shell**

man pages

- Negli script di shell sono spesso utilizzati comandi di sistema
  - ad esempio `ls, cp, cd, test` …
- Descrizione dettagliata di tutti i comandi di sistema nelle **man pages**
  - es. `man ls`
- Descrizione di tutte le feature della shell (**bash**)
  - `man bash`
  - `http://www.gnu.org/software/bash/manual/bashref.html`(Ottima guida)

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 direttorio **`dir`** e dovrà stamparne un sommario del contenuto su 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 una directory**: riportare il nome del direttorio ed il numero di direttori o 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 (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 "$i": `ls -1 "$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 "$i": `ls -1 "$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 "$i": `ls -1 "$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
```

Esercizio 2 – 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

`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 - Base

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

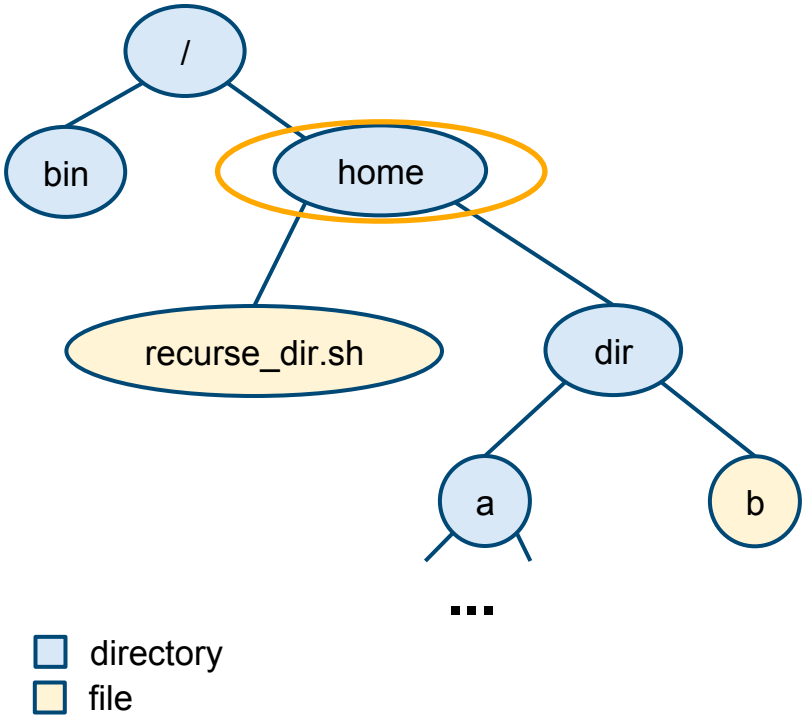
Caso base

Caso generale

Chiamata ricorsiva

Ricorsione - Un'espansione (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
exit 0
```

VARIABILI:

$PWD /home

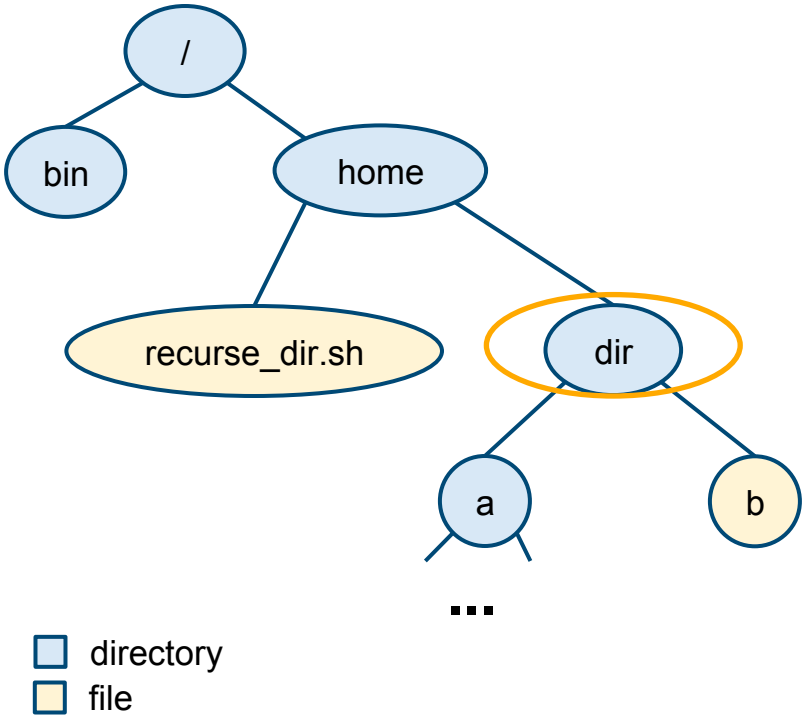
$0 /home/recurse_dir.sh

$1 dir

Ricorsione - Un'espansione (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
exit 0
```



VARIABILI:

$PWD /home/dir

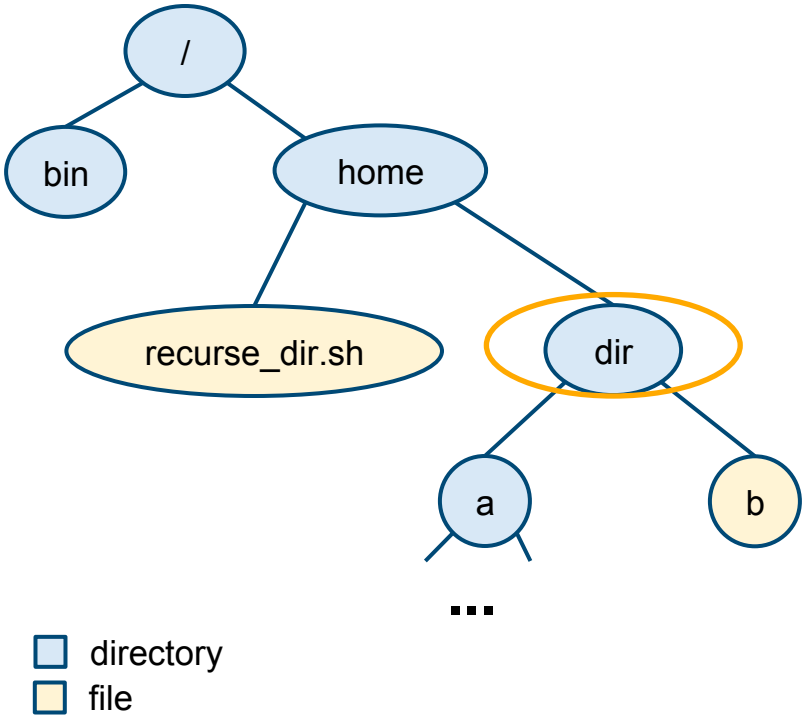
$0 /home/recurse_dir.sh

$1 dir

Ricorsione - Un'espansione (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
exit 0
```



VARIABILI:

$PWD /home/dir

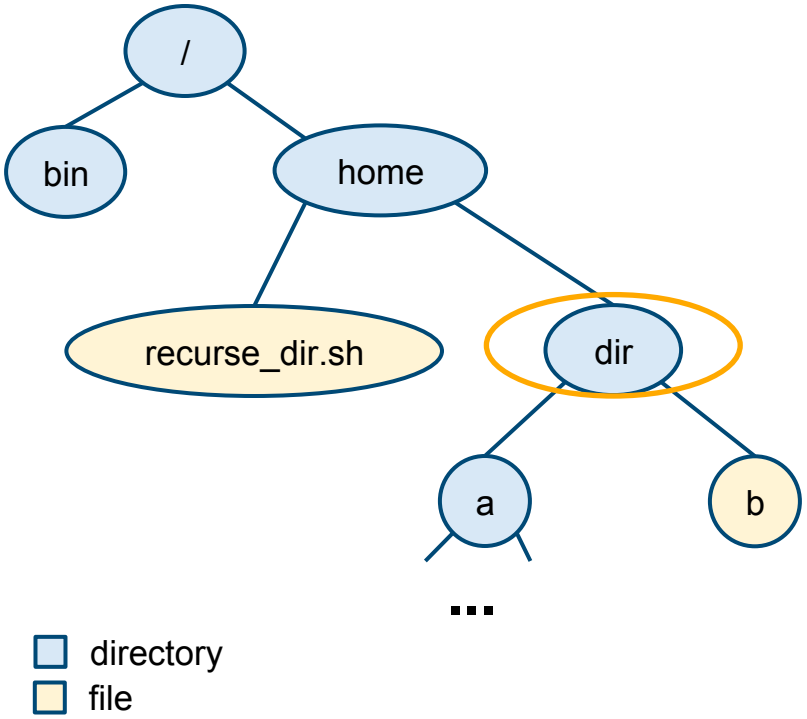
$0 /home/recurse_dir.sh

$1 dir

Ricorsione - Un'espansione (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
exit 0
```



VARIABILI:

$PWD /home/dir

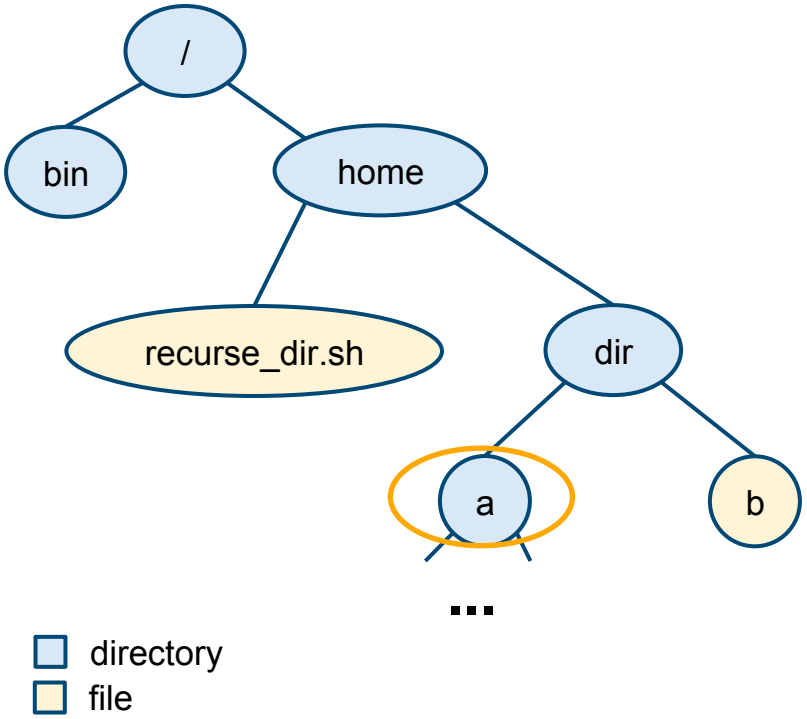
$0 /home/recurse_dir.sh

$1 **a**

Ricorsione - Un'espansione (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
exit 0
```



VARIABILI:

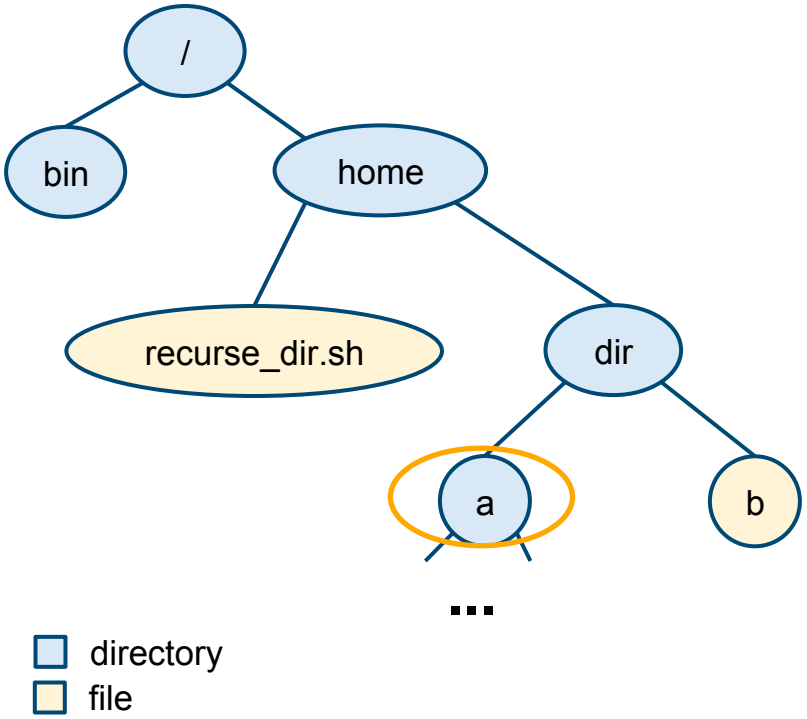
$PWD /home/dir/a

$0 /home/recurse_dir.sh

$1 a

Ricorsione - Un'espansione (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
exit 0
```

VARIABILI:

$PWD /home/dir/a

$0 /home/recurse_dir.sh

$1 ..

ATTENZIONE

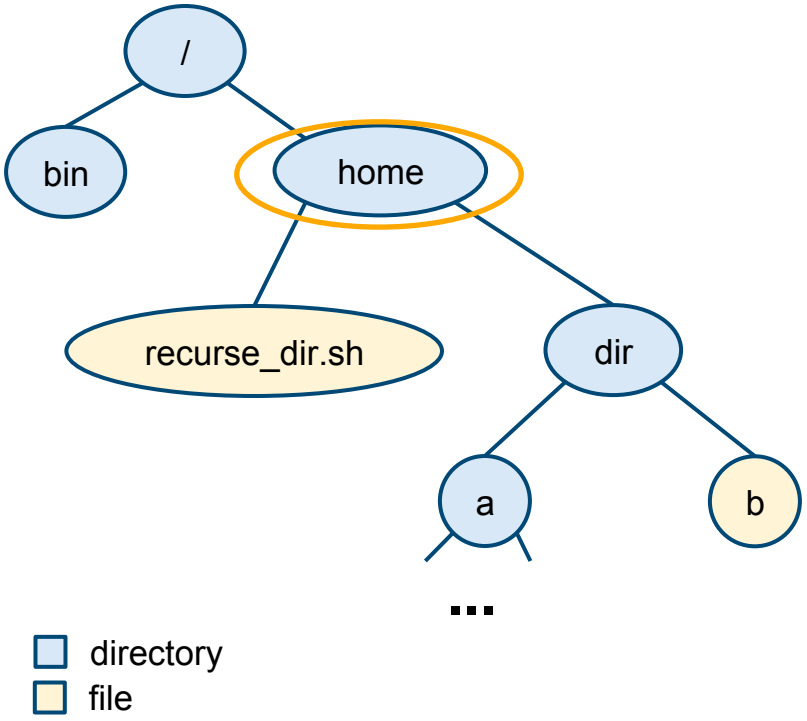
Nell'esempio lo script è stato invocato specificando il suo path assoluto!

Cosa succederebbe invocandolo con un path relativo?

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

Ricorsione - Un'espansione alt.(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
exit 0
```

VARIABILI:

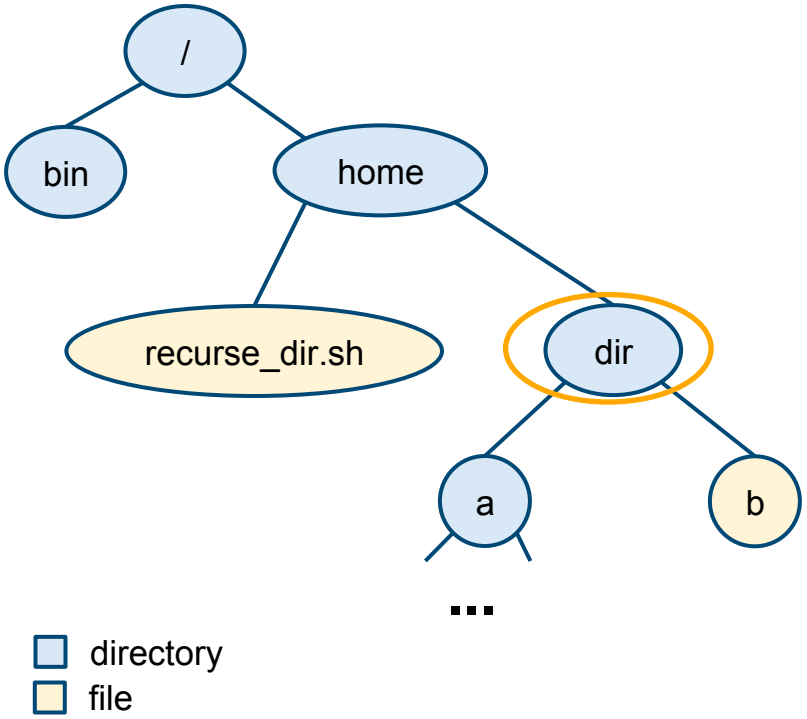
$PWD /home

$0 **./recurse_dir.sh**

$1 dir

Ricorsione - Un'espansione alt.(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
exit 0
```

VARIABILI:

$PWD /home/dir

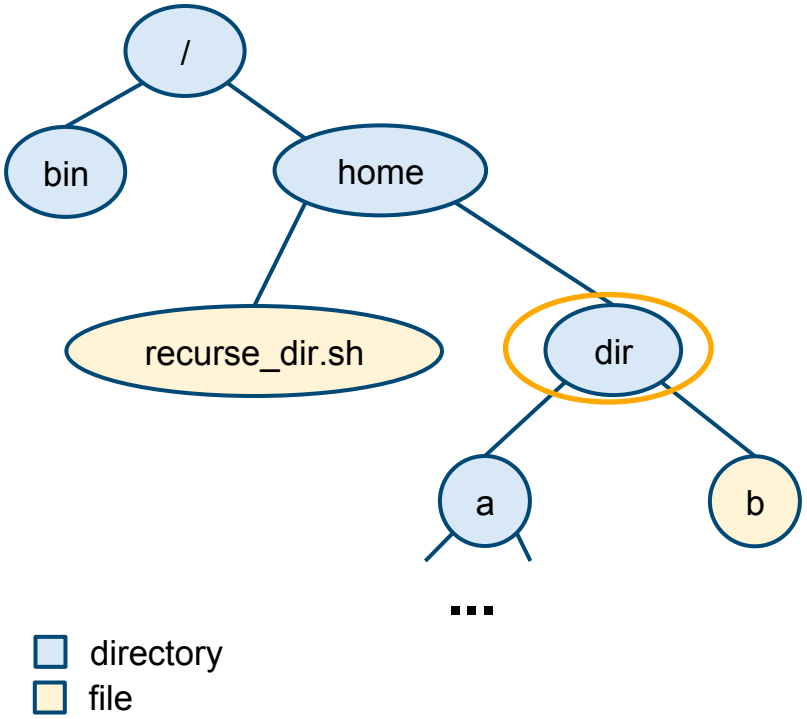
$0 ./recurse_dir.sh

$1 dir

Ricorsione - Un'espansione alt.(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
exit 0
```



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

- **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

Script di invocazione

recurse_dir.sh

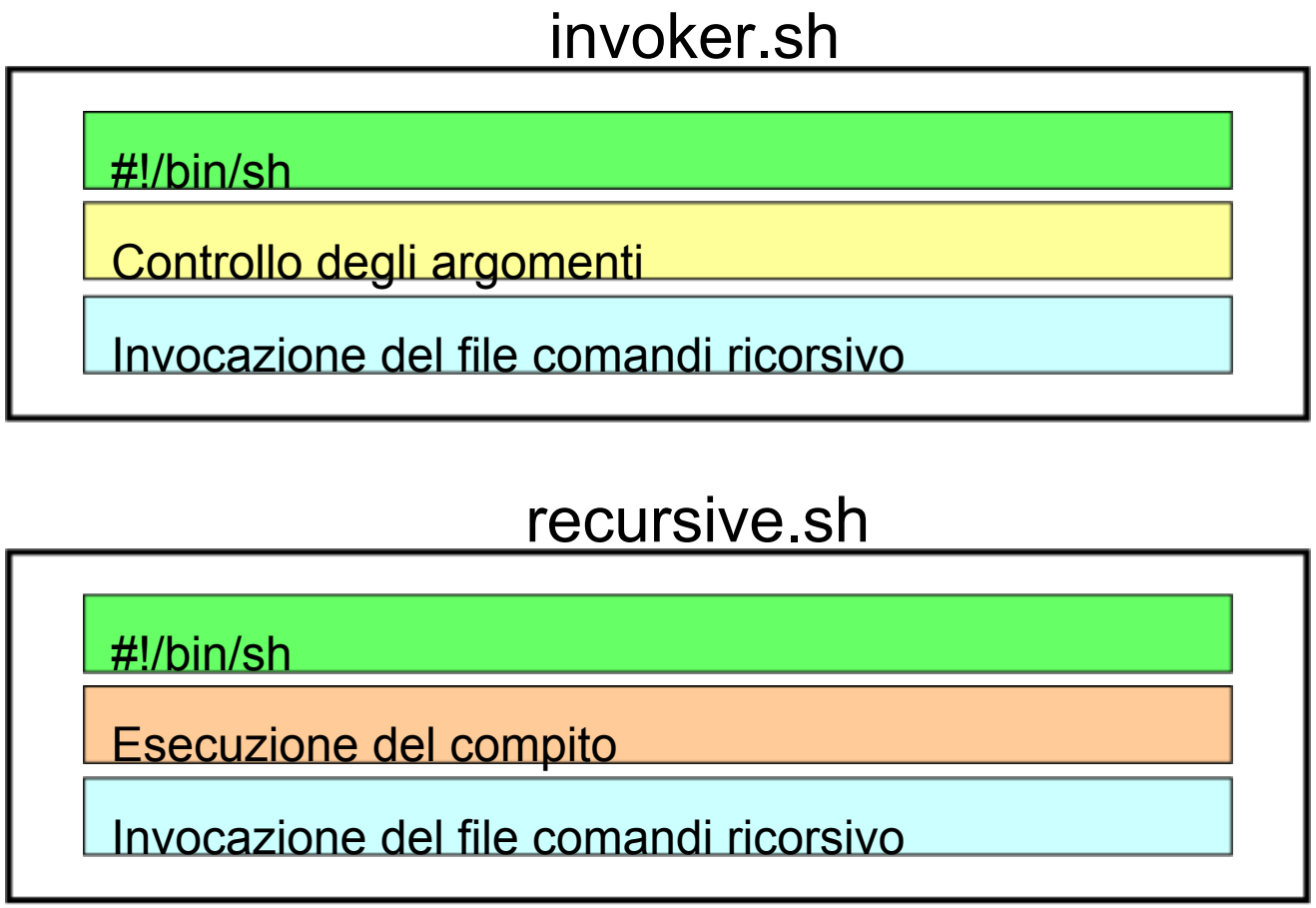
```
#!/bin/bash
# ... controllo argomenti

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
exit 0
```

Nota: stiamo supponendo che lo script **recurse_dir.sh** venga sempre invocato dalla directory che lo contiene. Es: se **recurse_dir.sh** e **do_recurse_dir.sh** sono in **/home/dloreti**, supponiamo che la directory corrente al momento dell'invocazione sia **/home/dloreti**

Struttura di un file comandi ricorsivo



Esercizio 3 – Esplorazione ricorsiva del file system (1/2)

Realizzare un file comandi (ricorsivo) che abbia la sintassi

```
search.sh minSize maxSize 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.

`minSize, maxSize` sono due interi

Esercizio 3 – (2/2)

Il compito del file comandi è quello di :

- Visitare (ricorsivamente) tutti i **sottoalberi** individuati da `dir1 … dirN`

- Per ogni file trovato verificare che la **dimensione in KB** sia compresa tra `minSize` e `maxSize`

  Suggerimento: vedere il comando **stat** per ottenere la dimensione del file (`man stat`)

- 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