Esercitazione 7
Gruppo AK

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

Caso base

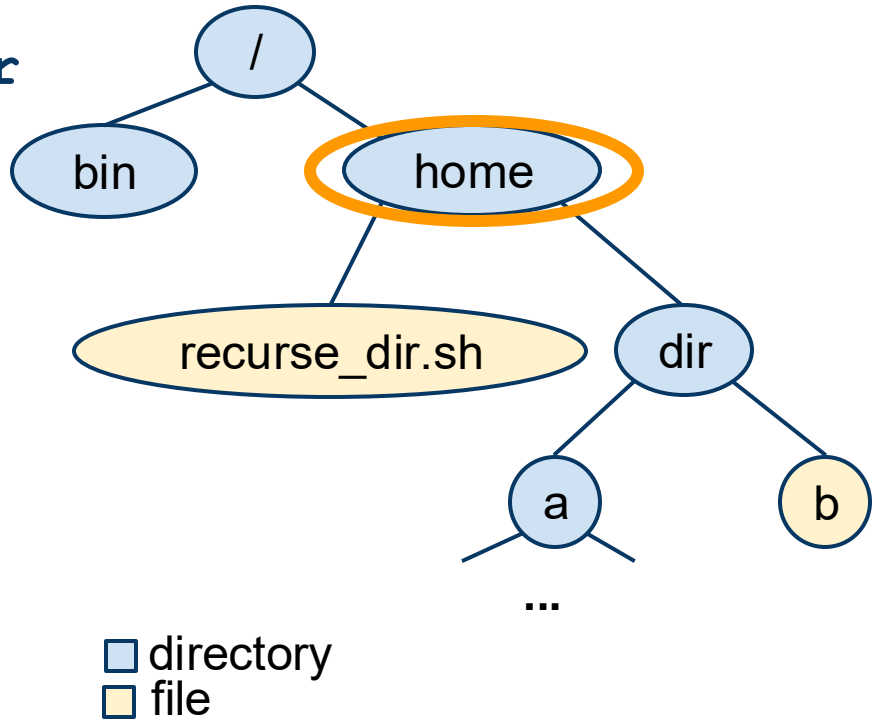
Caso generale

Chiamata ricorsiva

Ricorsione (1/6)

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

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



VARIABILI:

$PWD /home

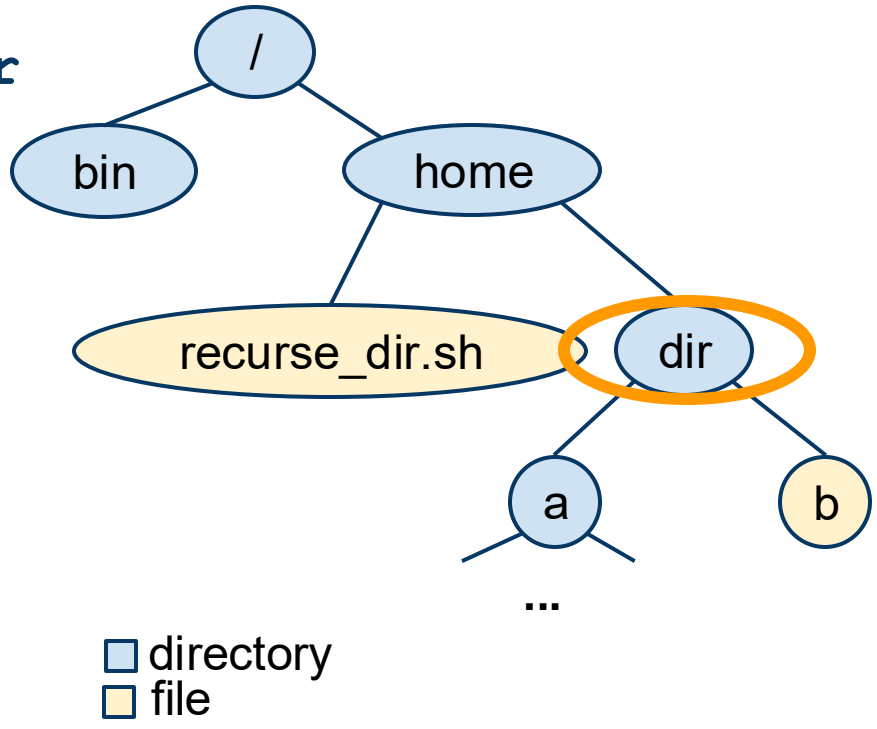
$0 /home/recurse_dir.sh

$1 dir

Ricorsione (2/6)

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

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



VARIABILI:

$PWD **/home/dir**

$0 /home/recurse_dir.sh

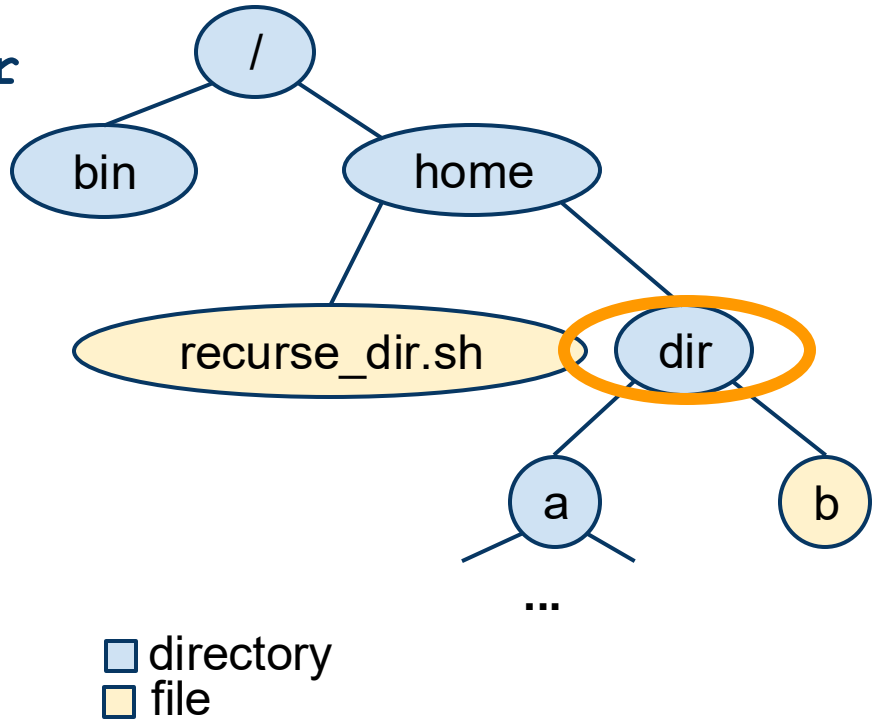
$1 dir

Ricorsione (3/6)

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

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

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



VARIABILI:

$PWD /home/dir

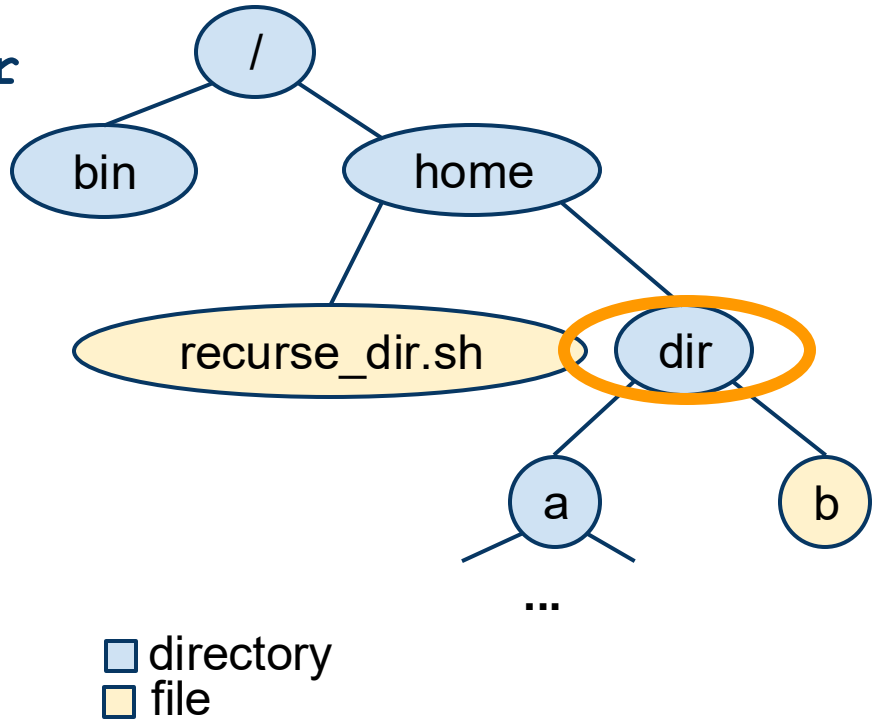
$0 /home/recurse_dir.sh

$1 dir

Ricorsione (4/6)

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

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



VARIABILI:

$PWD /home/dir

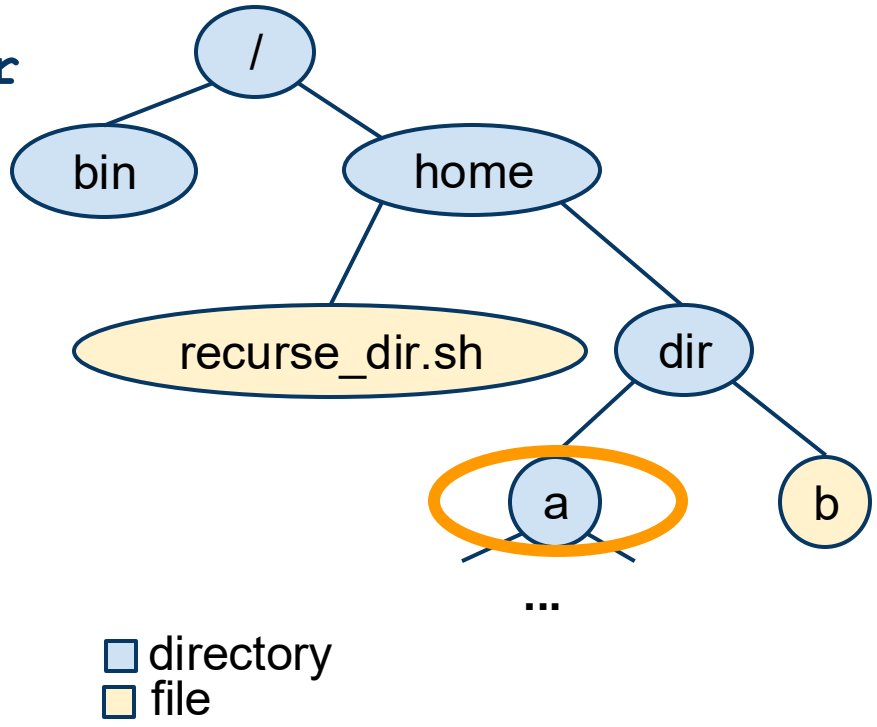
$0 /home/recurse_dir.sh

$1 **a**

Ricorsione (5/6)

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

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



VARIABILI:

$PWD **/home/dir/a**

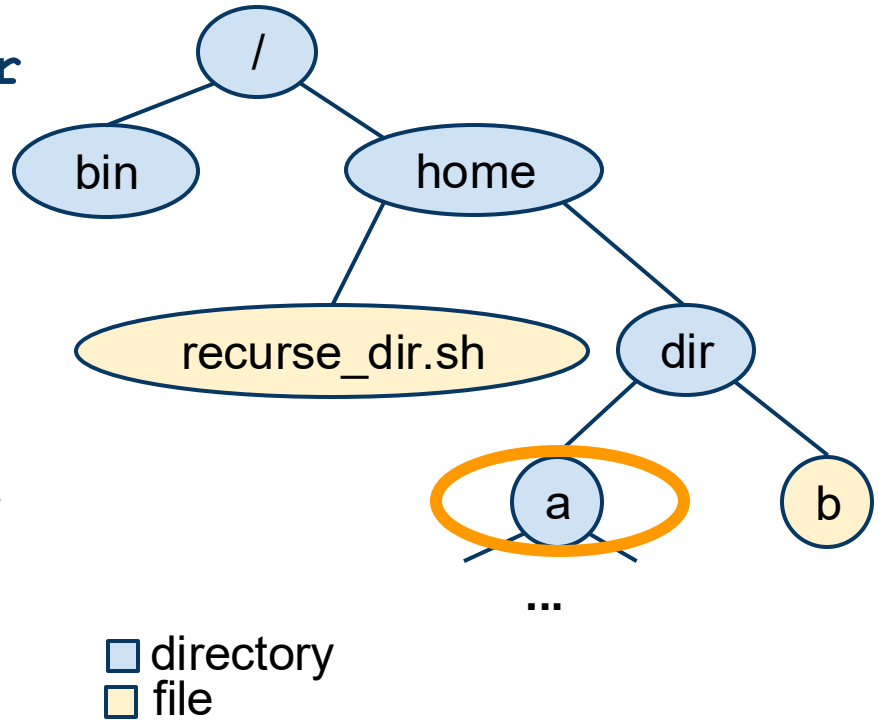
$0 /home/recurse_dir.sh

$1 a

Ricorsione (6/6)

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

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



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
```

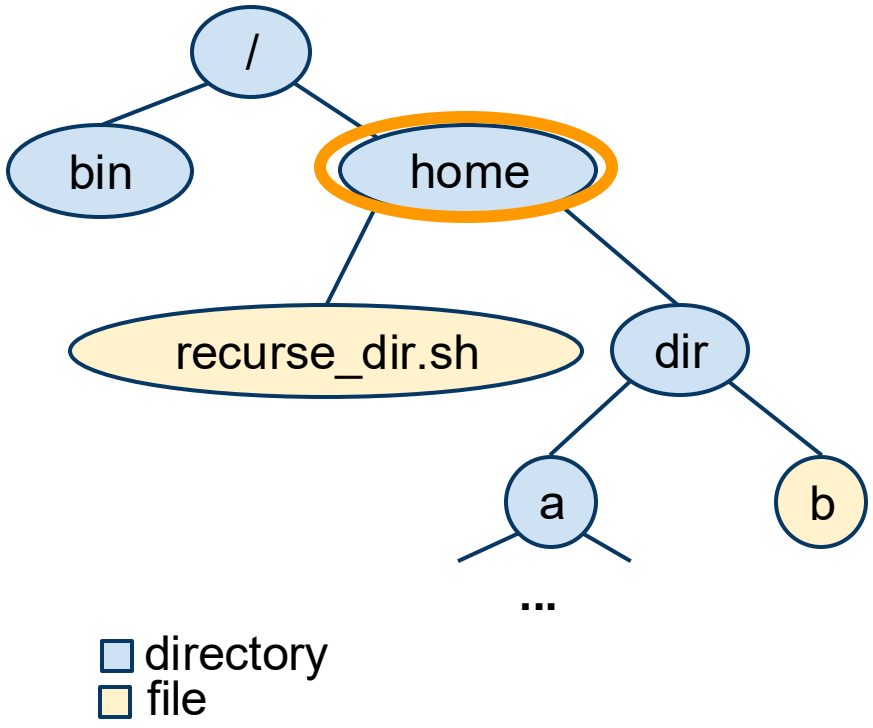
Cosa succederebbe invocandolo
con un **nome relativo**?

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

Ricorsione - alternativa (1/3)

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

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



VARIABILI:

$PWD /home

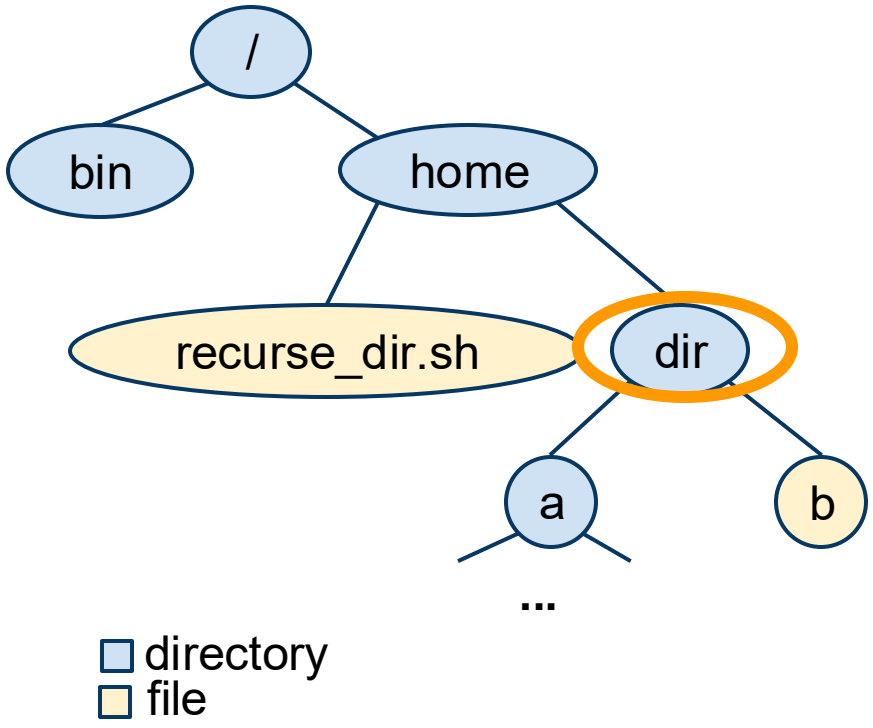
$0 **./recurse_dir.sh**

$1 dir

Ricorsione - alternativa (2/3)

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

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



VARIABILI:

$PWD **/home/dir**

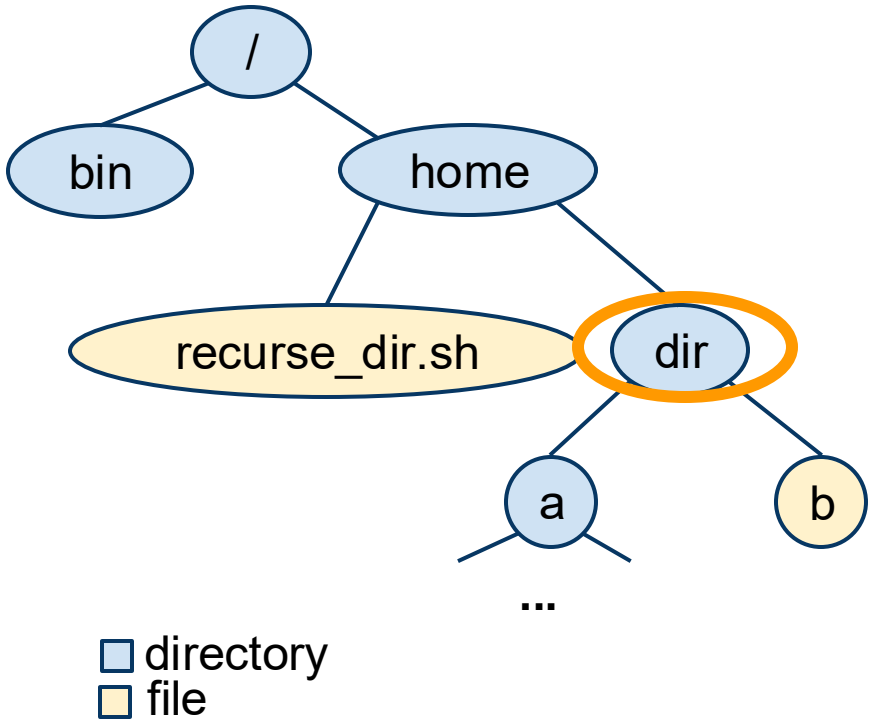
$0 ./recurse_dir.sh

$1 dir

Ricorsione - alternativa (3/3)

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

```
$ ./recurse_dir.sh  a
if test -f "$1" ; then
    echo `pwd`/$1
elif test -d "$1" ; then
    cd "$1"
    for f in * ; do
        "$0" "$f"
    done
fi
```



non funziona!

VARIABILI:

$PWD /home/dir

$0 ./recurse_dir.sh

$1 dir

Come risolvere?

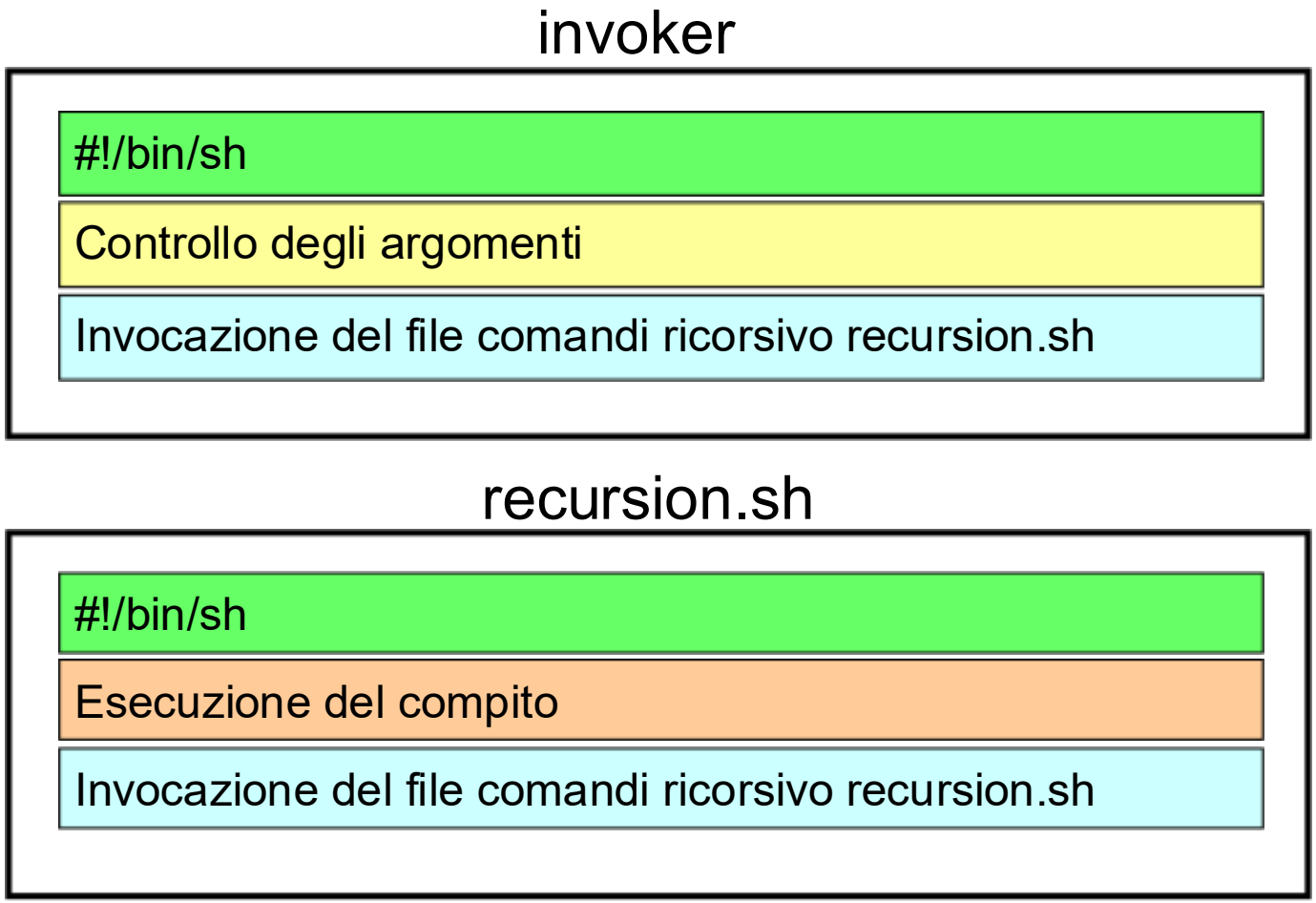
Problema: Un valore dipendente dalla directory di lavoro corrente (un percorso relativo) viene "**propagato**" da una invocazione ricorsiva all'altra (tramite la variabile **`$0`**)
La directory di lavoro però cambia (perchè usiamo il comando **`cd`** nel codice)

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



Soluzione naïve: sfruttiamo la variabile **PATH**

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**.

Invoker: **recurse_dir.sh**

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

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

recursion.sh

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

Soluzione naïve: problema

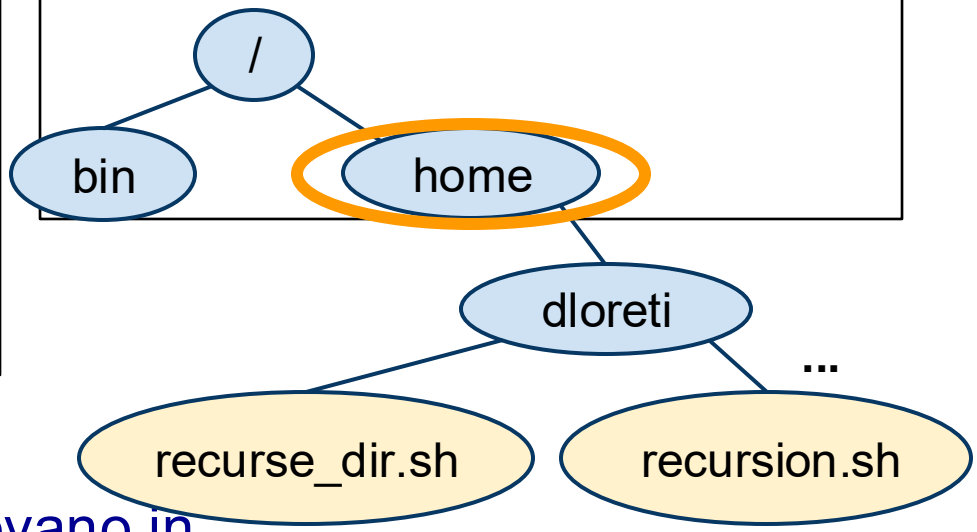
Invoker: **recurse_dir.sh**

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

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

recursion.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**

=> **recursion.sh** viene cercato in **/home** → NON TROVATO!

Soluzione generale (1/4)

Invoker: `recurse_dir.sh`

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

if [[ "$0" = /* ]] ; then
  # se $0 è un path assoluto
  dir_name=`dirname "$0"`
  recursive_cmd="$dir_name/recursion.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/recursion.sh"
else
  # Non si tratta nè di un path relativo, nè di uno
  # assoluto, il comando $0 sarà cercato in $PATH.
  recursive_cmd=recursion.sh
fi
#Invoco il comando ricorsivo
"$recursive_cmd" "$1"
```

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

Lancia `recursion.sh`

```
#!/bin/bash
if …
```

Soluzione generale (2/4)

Invoker: **recurse_dir.sh**

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

if [[ "$0" = /* ]] ; then
  # se $0 è un path assoluto
  dir_name=`dirname "$0"`
  recursive_cmd="$dir_name/recursion.sh"
elif [[ "$0" = */* ]] ; then
  # se c'è un
  # $0 è un p
  dir_name=`d
  recursive_c
else
  # Non si tratta né di un path relativo, né di uno
  # assoluto, il comando $0 sarà cercato in $PATH.
  recursive_cmd=recursion.sh
fi
#Invoco il comando ricorsivo
"$recursive_cmd" "$1"
```

Esempio:
$0=/home/recurse_dir.sh
`dirname "$0"` ➔ /home
$recursive_cmd=/home/recursion.sh

Soluzione generale (3/4)

Invoker: **recurse_dir.sh**

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

if [[ "$0" = /* ]] ; then
  # se $0 è un path assoluto
  dir_name=`dirname "$0"`
  recursive_cmd="$dir_name/recursion.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/recursion.sh"
else
  # Non
  # asso
  recurs
fi
#Invoco
"$recursive_cmd" "$1"
```

Esempio:
$0=../folder/recurse_dir.sh
`dirname "$0"` ➔ ../folder
$recursive_cmd=/home/../folder/recursion.sh

Soluzione generale (4/4)

Invoker: **recurse_dir.sh**

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

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

Esempio:
$0=recurse_dir.sh
$recursive_cmd=recursion.sh

Esercizio 1 (1/2)

Realizzare un file comandi (ricorsivo) che abbia la sintassi

```
contaRigheOcc dir strToFind
```

dove:

- **`dir`** è il nome assoluto di una directory esistente nel file system
- **`strToFind`** è una stringa

Il file comandi deve innanzitutto controllare che i parametri passati siano corretti

Esercizio 1 – (2/2)

Il file comandi deve esplorare **ricorsivamente** il sottoalbero individuato da **`dir`** e, per ogni file **ordinario,** contare il numero **X** di righe con almeno una occorrenza della stringa **`strToFind`**

- Se X è pari, stampare a video la stringa:

  Il file <absNameFile> contiene un numero pari (<X>) di righe rilevanti

  Dove:

  - <*absNameFile*> è il nome assoluto del file
  - <*X*> è il numero (pari) di righe con almeno una occorrenza di **`strToFind`** in <*absNameFile*>

- Se X è dispari, stampare una analoga stringa su un file **report.log** nella home dell'utente che ha invocato lo script.

Esercizio 1: Suggerimenti (1/2)

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

Poi:

- Contare il numero di righe in un file con almeno una occorrenza di una stringa → usare **l'opzione apposita del comando `grep` (vedere il man).**
- Controllo se X è pari o dispari→controllare gli operatori possibili nel man di **`expr`**.

Esercizio 1: Suggerimenti (2/2)

- Se X è dispari, dobbiamo scrivere su **report.log.**
  - La stampa deve essere fatta aggiungendo una riga per ogni file con X dispari
  - Cosa succede se il file esiste già al momento dell'invocazione? Qualora esista, assicurarsi anche che sia vuoto all'inizio dell'esecuzione.

- <*absNameFile*> deve riportare il path assoluto del file.
  - Lo script userà `cd` per saltare nel direttorio (e poi analizzare ogni file in esso contenuto)
  - l'absolute name del file può essere ricostruito sulla base del direttorio corrente

Esercizio 2 (1/2)

Realizzare un file comandi (ricorsivo) che abbia la sintassi

`minOcc H strToFind dir1…dirN`

dove:

- **`H`** è un intero positivo
- **`strToFind`** è una stringa
- **`dir1…dirN`** sono nomi assoluti di direttori esistenti nel file system

Dopo aver controllato il corretto passaggio dei parametri in ingresso, il file comandi deve eseguire una ricerca in ciascun sottoalbero individuato dalle directory **`dir1…dirN`** .

Esercizio 2 (2/2)

Per ogni directory (o sotto-direcory) **`dir_i`** :

- Analizzare tutte e sole le prime **`H`** righe di tutti i regular file in **`dir_i`** e determinare il numero ***Y*** di occorrenze di **`strToFind`**
- Determinare il file in **`dir_i`** per cui ***Y*** è minimo
- Stampare sullo stdout una riga di riepilogo del tipo:

  <AbsNameDir_i> <fileMin> <YMin>

  Dove:

  - *<absNameDir_i>* è il nome assoluto di **`dir_i`**
  - *<fileMin>* è il basename del file col minor numero di occorrenze di **`strToFind`** nelle prime **`H`** righe
  - *<YMin>* è il corrispondente numero (minimo) di occorrenze

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
- Filtrare solo le prime **`H`** righe di ogni file → opportuna opzione del comando **`head`**
- Contare il numero di occorrenze di una stringa in un file → controllare nel man l'opzione **`-o`** di **`grep`**. Testare l'out a linea di comando. Come funziona? Come usarla per contare le occorrenze?