# Supponiamo ....

- Di usare split per spezzare una sequenza in sottosequenze separate da istanze di una espressione regolare
- Di voler uteriormente elaborare le sottosequenze reperite ... (ad esempio stampare solo quelle che hanno un certo matching e contarle)

# Soluzione 1

- Scrivo un programma unico, ad esempio:

```
BEGIN {counter=0;}
{
   split($0, subseq, /(ab)+/);
   for (i in subseq)
        if (subseq[i] ~ /b(a|c)b/) {
           print subseq[i];
           counter++;
           }
}
END {print “totale = ” counter
```

# Soluzione 2 ...

- Scrivo due programmi:

```
{
  split($0, subseq, /(ab)+/);
  for (i in subseq) print subseq[i];
}
```

spezza

```
BEGIN {counter=0;}
subseq[i] ~ /b(a|c)b/ {
                        print subseq[i];
                        counter++;
}
END {print “totale = ” counter}
```

elabora

# ... Soluzione 2

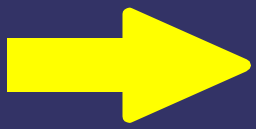
- gawk -f spezza dati_iniziali > output1
- gawk -f elabora output1
- rm output1

- Vantaggio: posso riutilizzare i due programmini pr eseguire elaborazioni diverse senza doverli modificare!

# Per usi frequenti della combinazione

- Posso creare uno script di shell!

- File di testo che contiene una sequenza di comandi
- Il file va reso eseguibile con chmod
- Poi può essere usato come un comando



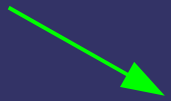
# Creazione dello script

```
gawk -f  spezza  dati_iniziali > output1
gawk -f  elabora  output1
rm output1
```

File:
spezzaElabora

- chmod a+x  spezzaElabora
- spezzaElabora

eseguo
spezzaElabora



rendo
spezzaElabora
eseguibile

# Shell UNIX

Bash
(Bourne Again Shell)

# Shell

- È l'ambiente di lavoro di ciascun utente
- Un utente ha tante shell quanti sono i terminali aperti con il suo login
- L'ambiente di lavoro può essere impostato attribuendo valori a certe variabili di sistema
- È possibile scrivere programmini (script) che compongono i comandi normalmente dati da linea di comando

# Script

- File di testo contenente comandi
- La prima linea può contenere l'indicazione del tipo di shell che deve eseguire quel codice, in tal caso si scrive nella prima linea, per esempio:

- #!/bin/csh

# Esempio

```
#!/bin/bash

# questo e' un commento
# listo il contenuto della directory in temp
ls > temp

# conto il numero di linee del file
echo `wc -l temp`

# rimuovo il temporaneo
rm temp
```

# Bash

- Esistono diversi tipi di shell che offrono funzionalità diverse: sh, csh, tcsh, ksh, pksh, bash, ...
- Bash è la shell di default dei sistemi Linux
- Il suo nome deriva da quello del ssuo ideatore, Steven Bourne

# Funzionalità di bash

- Completamento automatico dei comandi
- Editing della linea di comando
- History
- Definizione di alias
- Definizione di variabili (di ambiente e di shell)

# Completamento automatico

- Supponiamo di voler applicare il comando cat al file DocIniziale_v31.tex e che non vi siano altri file (nella directory di lavoro) aventi nome che inizia per 'Do'. Se digito:

- > cat  DoTAB

- Dove TAB indica il tasto di tabulazione, il nome del file verrà completato automaticamente. In caso di ambiguità viene emesso un segnale acustico

# Editing della linea di comando

- È possibile percorrere la linea di comando usando le frecce destra e sinistra
- È possibile modificarne il contenuto cancellando caratteri con backspace e canc
- È possibile inserire nuovi caratteri in posizioni arbitrarie
- Se, per esempio digito erroneamente “ls Docimenti” anziché “ls Documenti” posso usare le frecce per spostarmi lungo la linea di comando e correggere l'errore

# History

- La shell memorizza gli ultimi N comandi eseguiti in una lista. Posso percorrere tale lista dalla linea di comando con le frecce ↑ e ↓
- Il comando history visualizza i comandi memorizzati, ciascuno preceduto da un numero d'ordine
- Per eseguire l'n-mo comando posso usare l'abbreviazione: !n
- Per eseguire il comando più recente: !!

# Definizione di alias

- È possibile rinominare comandi o definrne di propri tramite il comando alias, per esempio:

```
alias listadoc='ls *.doc'
```

- listadoc potrà essere usato come un normale comando. Il suo effetto è listare tutti i file con estensione doc della working directory

# Definizione di variabili

- La definizione di una variabile di shell avviene con l'assegnazione di un valore (es. Da linea di comando):

  mia_var=val

- mia_var è il nome della variabile, val il valore.
- Una volta definita una variabile può essere usata nella shell facendone precedere il nome da $

- mia_dir=~/progetti/2004/programmi
- echo $mia_dir
- ~/progetti/2004/programmi
- ls $mia_dir
- ...

# Variabili speciali

- Sono predefinite e consentono di configurare l'ambiente di lavoro; non si aggionrano da linea di comando ma all'interno dei file di configurazione
- Fra queste: PATH, HOME, USER, PS1, ...
- PATH: contiene un insieme di cammini (path, per l'appunto) all'interno dei quali la shell cerca i comandi o i programmi di cui è richiesta l'esecuzione. Il valore di PATH è multiplo (lista di valori) va normalmente aggiornato quando si installa un nuovo software
- PS1: configura il prompt. Ha valore singolo, che può essere modificato a piacere

# File di configurazione

- La configurazione di una shell avviene tramite un insieme di file di testo nascosti, che gli utenti possono modificare a piacere per adattare l'ambiente di lavoro alle proprie esigenze
- Per le bash i file sono:

  .bash_profile
  .profile
  .bashrc

- Vanno toccati con cautela!!! Meglio farne una copia di backup prima di modificarli

# Esperimento ...

- Provate a digitare su linea di comando:

- PS1=”comandi> “

- Che cosa succede?
- Provate ora a creare una nuova shell
- Che aspetto ha il prompt?

# ... morale

- Se modifico il valore di una variabile di sistema da linea di comando, la nuova configurazione sarà valida solo nella finestra corrente!
- Come si fa a rendere le impostazioni pervasive?
- Occorre saperne di più sulle shell ...

# Generazione di bash/shell

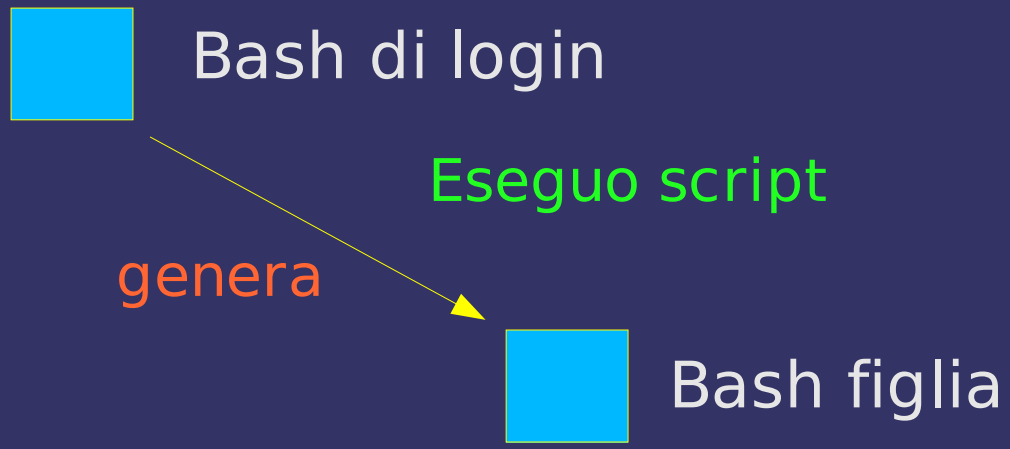
- All'atto del login viene generata una shell che servirà l'utente
- Ogni volta che viene eseguito uno script la shell di lavoro genera una sub-shell (shell figlia) il cui compito è eseguire lo script
- Una shell figlia può a sua volta generare shell figlie
- Conseguenza: in generale anche su di uno stesso terminale possono coesistere tante shell contemporaneamente, in modo trasparente all'utente

# Esempio

- Quando l'utente da linea di comando esegue lo script spezzaElabora visto prima, la shell che fa comparire il prompt a video genera una shell figlia che rimane in vita il tempo necessario ad eseguire lo script e poi termina

# Bash figlie e variabili

- Se io creo una variabile in una shell, questa sarà visibile anche all'interno delle sue figlie?




- In generale no
- Sono visibili solo le variabili esportate

# Eredità di variabili

- Una bash figlia eredita una copia delle variabili esportate dalla bash madre, con il valore che avevano al momento della sua generazione
- Le due copie sono indipendenti
- Le variabili esportate sono dette di ambiente
- Per esportare una variabile si usa export:

```
export var_da_esportare
```