ESEMPIO:

Si vuole la possibilità di dare argomenti con formato più 'libero': come specificare il mese genn o Jan, etc.

#calend [mese [anno]]

```
case $# in
0) set ‘date‘; m=$2; y=$6;;
# ottiene gli argomenti da date come $1, $2, $3 etc.
1) m=$1; set ‘date‘; y=$6;;
*) m=$1; y=$2;;
esac

case $m in
  jan* |Jan* |Gen* |gen*)          m=1;;
  f*|F*)                          m=2;;
  mar*|Mar*)                      m=3;;
  ap*|Ap*)                        m=4;;
  mag*|Mag* |May | may)           m=5;;
  gi*|Gi* |Jun* |jun*)            m=6;;
  lu*|Lu* |Jul* |jul*)            m=7;;
  au*|Au* |Ag* |ag*)              m=8;;
  s*|S*)                          m=9;;
  o*|O*)                          m=10;;
  n*|N*)                          m=11;;
  d*|D*)                          m=12;;
esac
/usr/bin/cal $m $y
```


ESEMPIO:

Il filesorgente viene copiato nel file destinazione:
se questo esiste già, si chiede conferma della sovrascrittura

#copia filesorg filedest

```
case $# in
0 | 1) echo Usage $0 filesorg filedest
       exit 1;;
2)  if test ! -f $1
        then   echo $1 non esiste;  exit 2
    fi;;
esac
if test -f $2
then   echo il file $2 esiste: vuoi ricoprirlo?
       read answer
       case $answer in
           n* | N* ) exit;;
       esac
fi
if test -d $2
    then echo $2 "è un direttorio";  exit 3
else  if cp $1 $2
       then echo ho copiato il file $1 nel file $2
       else echo errore sulla copia
  fi
fi
```

ESEMPIO:

Il comando muove su richiesta tutti i file passati come argomento in un direttorio specificato

#moveat dir file1 file2 ... filen

```
case $# in
 0|1) echo almeno due argomenti; exit 1;;
esac
if test -d $1
then
 for i in $* # per tutti gli argomenti
 do
    if test $i != $1
    then
        echo $i in $1 ?
        read risposta
        if test $risposta = "si" -o $risposta = "s"
          then
            if  mv $i $1
  # il risultato della mv pilota la alternativa
                then echo si $1/$i
                else echo problemi per $1
            fi
        fi
    fi
done
fi
```


ESEMPIO:

Si inseriscono in un file (il cui nome è dato come parametro), senza distruggerlo, stringhe lette da input se soddisfano certe condizioni

#insert file

```
while  echo "vuoi finire(si/no)?"; read fine
       test $fine != si
do
  echo inserisci stringa
  read a
  echo $a
  case $a in
  ?[b-g]*a)                 echo OK1; echo $a >> $1;;
  ?s*[0-9]? | [0-9]s)       echo OK2; echo $a >> $1;;
  ??o*[2-9])                echo OK3; echo $a >> $1;;
  ??[A-Z]*r? | ?r[A-Z])     echo OK4; echo $a >> $1;;
  *)                        echo NO case;;
  esac
done
```

ESEMPIO:

Si vuole fare il monitoring degli utenti presenti nel sistema. In particolare si usa il comando diff tra la lista degli utenti ottenuta in momenti successivi

```
#match
new=/usr/tmp/who1$$
old=/usr/tmp/who2$$
> $old
while :        # ciclo infinito
do
   who > $new
   diff $old $new
   echo move
   mv $new $old
   sleep 60  # ritardo di un minuti tra i cicli
done
```

Cosa vale $$ per ogni invocazione?
Come invocare il comando **match**?

confronta:
match
match &

Alcuni semplici filtri

```
while read line
do
  echo $line
done
```

```
while read line
do  ;  case $line in
       *[0-9]*) echo $line ;;
       esac
done
```

```
while read line
do
 echo $line > /dev/tty
 if chiedi "Da copiare? " ; then echo $line;  fi
done
```

Componenti software

```
cat fx | while read line ; do
  case $line in
    *[0-9]*) echo $line ;;
  esac
done | wc -l
```

```
cat fx | while read line
do
  echo $line > /dev/tty
  if chiedi "Da copiare? "
     then echo $line
  fi
done > fy
```

ESEMPIO:

Si vuole verificare la presenza di un dato file in tutti i sottodirettori del path attuale

```
#where file
case $# in
    0) echo almeno un argomento; exit 1;;
    esac
for i in ‘echo $PATH | tr ’:’ ’ ‘
do
     if test -f $i/$1
      then echo $i/$1
     fi
done
```

Si estenda il file comandi in modo da fornire solo il primo sottodirettorio in cui si trova un file del nome dato

Questa funzione è parte dello shell quando si deve mettere in esecuzione un nuovo file.

ESEMPIO:
Si cercano i file (con il nome specificato), che siano leggibili o scrivibili, nella gerarchia a partire dal direttorio specificato

```
#cerca dir file
case $# in
0|1) echo errore Usage $0 dir file
     exit 1;;
2) ;;
esac

PATH=$PATH:/usr/antonio/provesh
# il file comandi si invoca ricorsivamente:
# come diventa il path in una invocazione innestata?

if test ! -d $1
   then echo errore: $1 non è un sottodirettorio;
        exit 2;
fi
cd $1
```

```
if test -f $2
then
  if test -r $2 -o -w $2
  then
      echo file $2; cat $2
      if test -r $2
        then echo file $2 almeno leggibile in ‘pwd‘
      fi
      if test -w $2
        then echo file $2 almeno scrivibile in ‘pwd‘
      fi
  fi
fi

for i in *
do
    if test -d $i
        then cerca $i $2
    fi
done
# l’invocazione di cerca con nome relativo richiede
# la estensione del PATH
```


ESEMPIO:
Si ricerca un file dato in una gerarchia.
La ricerca avviene in modo ricorsivo a partire da un sottodirettorio specificato

```
#dove dir nomefile
PATH=.:/bin:/usr/bin:/usr/antonio/prove
    # direttorio di residenza di dove

if test $# -lt 2
  then  echo almeno due argomenti; exit 1
fi
cd $1
# si dovrebbe controllare che l’argomento
# sia un sottodirettorio
for i in *
do
  if test -d $i
    then dove $i $2
  fi
done
if test -f $2
    then echo in ‘pwd‘: $2
fi
```

Si veda la funzionalità analoga del comando **find**.

ESEMPIO:
Si visualizza il contenuto dei file passati come argomento richiedendo per ciascuno l'assenso all'utente

#choose file1 file2 file3 ... filen

```
for i
do
  echo -n "$i si/? "  > /dev/tty
  read answer
  case $answer in
   y*|Y*|s*|S*) cat  $i;;
  esac
done
```

Si provino invocazioni come:

choose file1 file2 file3

choose .*

choose p*

È possibile riusare choose in pipe? per esempio:
choose *.c | grep string

ESEMPIO:
Valutazione di un comando passato come parametro in tutti i sottodirettori di un direttorio specificato.

#commric direttorio comando

```
PATH=/bin:/usr/bin:/usr/ucb:/usr/antonio/prove
cd $1
echo Eseguo la eval del comando $2 con 'pwd'
eval $2
for i in *
do
  if test -d $i
  then echo !!!! Nuovo Direttorio $i
       commric $i "$2"
  fi
done
```

Si provino comandi che non richiedono sostituzioni, come:
commric / ls -lga

e quelli che le richiedono, come:
commric / "ls -lga p*"

Perché si usano gli " ... " ?

ESEMPIO: cercare il file normale file1 nella gerarchia che ha come radice il direttorio dir1, se specificato, oppure il direttorio corrente

```
#cerca [dir1] file1
case $# in
  1)        ;;
  2)        if test ! –d $1 ; then
              echo "$1 non e' un direttorio" >&2 ; exit 1
            fi
            cd $1
            shift
            ;;
  *)    echo "Uso $0 [direttorio] file" >&2 ; exit 1 ;;
esac

PATH=/home/antonio/shell:$PATH

if test -f $1
  then echo "Il file $1 e' in `pwd`"
fi

for i in *
do
  if test -d $i
     then cerca $i $1
  fi
done
```


Esempio: elencare i nomi di tutti i file contenuti nella gerarchia che ha come radice il direttorio dir1, se specificato, oppure il direttorio corrente

```
#collect [dir1]
case $# in
  0)    ;;
  1)    if test ! -d $1; then
          echo "$1 non e' un direttorio" >&2 ; exit 1
        fi
        cd $1 ;;
  *)    echo "Uso: collect [direttorio]" >&2 ; exit 1;;
esac
PATH=/home/antonio/shell:$PATH
ls
for i in *; do
  if test -d $i; then  collect $i; fi
done
```

Oppure, come prima, *senza ripetizioni*

```
#nomiunici dirName
old=""
collect $1 | sort | while read new; do
  if test "$new" != "$old"; then
    echo $new
    old=$new
  fi
done
```

oppure, meglio due soluzioni veloci

```
collect $1 | sort | uniq
collect $1 | sort -u
```

Esempio: elencare i nomi **assoluti** di tutti i file contenuti nella gerarchia che ha come radice il direttorio dir1, se specificato, oppure il direttorio corrente

```
#collect1 [dir1]
case $# in
  0)    ;;
  1)    if test ! -d $1; then
        echo "$1 non e' un direttorio" >&2 ; exit 1
        fi
        cd $1
        ;;
  *)    echo "Uso: collect [direttorio]" >&2 ; exit 1;;
esac

PATH=/home/antonio/shell:$PATH

for i in *; do
  echo `pwd`/$i
  if test -d $i; then
    collect1 $i
  fi
done
```


ESEMPIO:

Tre file comandi (trova, cerca e ritrova) consentono di trovare nei direttori specificati, tutti i file appartenenti ad un dato utente

```
#trova utente dirnames
case $# in
 0|1) echo usage is $0 utente dirnames >&2; exit 1;;
 *) ;;
esac
PATH=/home/antonio/bourne:$PATH;
UTENTE=$1;
cd /home
export UTENTE
export PATH
shift
cerca $*
```

Come regolarsi in caso non si voglia estendere il PATH?

#cerca dirnames

```
for i in *
do
 if test -d $i
  then  # echo $i $*
    for j in $*
     do
       if test $i = $j
          then cd $i;  ritrova $*; cd ..
          else cd $i;  cerca $*; cd ..
       fi
      done
  fi
done
```

#ritrova dirnames

```
ls -lga | grep $UTENTE > /usr/tmp/file
echo ritrovato ‘pwd‘
cat /usr/tmp/file
rm /usr/tmp/file
cerca $*
```


ESEMPIO:
Si ricercano i nomi unici di tutti i file appartenenti ad una data gerarchia

#nomiunici dir

```
echo $0 $1
case $#  in
 0) echo errore un parametro; exit 1;;
 1|*) if test ! -d $1
         then   echo errore
                exit 2;
       fi;;
esac

> $1/temp
colleziona $1 $1/temp
# il file comandi inserisce nel file specificato dal
# secondo argomento tutti i nomi dei file di una
# gerarchia (specificata nel primo argomento)

sort < $1/temp > $1/temp1
cat $1/temp1
> $1/temp;
old1=cicicicicicici
old2=cocicicicicici
```

```
for i in ‘cat $1/temp1‘
do
   # echo old2 $old2 old1 $old1 i $i
   if test $old1 != $i -a $old2 != $old1 -a
            $old1 != cicicicicicici
        then echo $old1 >> $1/temp
   fi
   old2=$old; old1=$i
done

if test $old1 != $old2
    then echo $old1 >> $1/temp
fi
```

#colleziona dir temp

```
PATH=$PATH:/usr/antonio/shell
cd $1
ls >> $2
for file in *
do
        if  test -d $file
            then colleziona $1/$file $2
        fi
done
```


ESEMPIO:
Si selezionano i nomi di file che compaiono almeno due volte nella gerarchia data

#filedoppi dir

```
echo $0 $1
case $#  in
 0) echo errore un parametro; exit 1;;
 1|*) if test ! -d $1
        then echo errore
        exit 2;
      fi;;
esac
colleziona $1 $1/temp
sort < $1/temp > $1/temp1
> $1/temp;
old1=cicicicicicici
old2=cococococococo
for i in ‘cat $1/temp1‘
do
   if test $old1 = $i -a $old2 != $i
            then echo $i >> $1/temp
   fi
   old2=$old1; old1=$i
done
```

ESEMPIO:
Due file comandi controllano le differenze dei nomi di file fra due gerarchie. L'obiettivo è listare i nomi dei file che sono presenti in una sola delle due gerarchie

 # diversi dir1 dir2

```
# lista file e direttori presenti in uno solo dei due arg
case $# in
0|1 )   echo Uso diversi dir1 dir2; exit 1;;
2)      if test ! -d $1 -o ! -d $2
        then echo due direttori; exit 2
        fi
        ;;
esac
PATH=$PATH:/home/antonio/shell
# aggiunge ad ogni invocazione
cd $1
ls $1 > temp1
ls $2 > temp2
diff temp1 temp2 > temp3
for i in `cat temp3`
do
 if test $i != temp1 -a $i != temp2
    then
     if test -f $1/$i
        then echo unico $1/$i
     fi
```



```
     if test -f $2/$i
        then echo unico $2/$i
     fi
     if test -d $1/$i
        then echo sottoalbero unico $1/$i; listaunici
     $1/$i
     fi
     if test -d $2/$i
        then echo sottoalbero unico $2/$i; listaunici
     $2/$i
     fi
fi
done
rm temp1 temp2 temp3

for dir in *
do  if test -d $1/$dir -a -d $2/$dir
        then diversi $1/$dir $2/$dir
        fi
done
```

```
# listaunici dir
# lista tutti i file e direttori

case $# in
0 ) echo Uso listaunici dir ; exit;;
1) if test ! -d $1
        then echo direttorio; exit
   fi;;
esac

PATH=/home/antonio/shell:.:/bin:/usr/bin:$HOME

cd $1
for i in *
do
  if test -f $i
     then echo unico $1/$i
        else if test -d $i
                then listaunici $1/$i
                fi
  fi
done
```


REVERSE

il filtro reverse deve operare sull'input
e fornire in uscita le linee in ordine inverso
(l'ultima per prima, ..., la prima per ultima)

I modo:
*uso di un file intermedio e togliendo una linea per volta (vedi **headm)***

```
# reverse nomefile
case $# in
1) ;; # ok
*)  echo Problemi: reverse ha un argomento; exit 1;;
esac

nome=/tmp/$1.$$
cp $1 $nome
while test ‘wc -l < $nome‘ -gt 0
do
    tail -1 $nome
    headmeno $nome 1 > $nome.$1
    mv $nome.$1 $nome
done
rm $nome
```

 # headm file

```
# toglie l'ultima linea dal file
case $# in
1)  ;;
*)  echo errore un argomento; exit 1;;
esac
v=‘wc -l < $1‘
v=‘expr $v - 1‘
head -$v $1
```

 # headmeno file argomentonumerico

```
# toglie dal file un certo numero di linee finali
case $# in
2)  ;;
*)  echo errore due argomenti; exit 1;;
esac
v=‘wc -l < $1 | tr -d ’ ’‘
if test $v -lt $2
 then  echo Errore: non ci sono le linee richieste
       exit 2;
fi
v=‘expr $v - $2‘
head -$v $1
```


II modo:

*uso di un file prodotto da **conta***
si mettono insieme i due file, linea a linea
poi si ordina secondo l'ordine numerico inverso
e si taglia ad ottenere il file iniziale rovesciato

 # reverse nomefile

```
x=‘wc -l < $1‘
conta $x > /tmp/$1.$1
paste -d: /tmp/$1.$1 $1 > /tmp/$1
sort -nr /tmp/$1 > /tmp/$1.$1
cut -d: -f2- /tmp/$1.$1
rm /tmp/$1 /tmp/$1.$1
```

#conta finoalnumero (da0alnumeroescluso)

```
x=0
while test $x -lt $1
do
echo $x
x=‘expr $x + 1‘
done
```

Evoluzioni
In UNIX si devono usare quanto possibile le pipe che non lasciano tracce e ridirezione
Sono i componenti elementari da usare

#numera (secondo necessità)

```
x=0
while read linea
do
  echo $x:$linea
  x=‘expr $x + 1‘
done
```

 # reverse

```
numera | sort -r | cut -d: -f2-
```

 # reverse

```
if read line
  then
        reverse
        echo $line
fi
```


FILTRO (in shell)

da sviluppare anche in C

si cercano in ogni linea dell’ingresso almeno un certo numero di argomenti diversi: se vengono trovati, la linea passa in uscita; altrimenti viene scartata.

filtro1 num c1 c2 … cN

num numero intero <= N
c1, **c2**, … **cN** caratteri alfabetici

```
#filtro1 num c1 c2 … cN
# filtro che richiede due argomenti e li usa
# il primo deve essere inferiore al numero degli altri
# gli altri sono caratteri
# si filtrano le linee che contengono almeno arg1
# caratteri  diversi

case $# in
  0|1)  echo Errore. Almeno 2 argomenti: n e car;
        exit 1;;
  *)   if test $1 -gt ‘expr $# - 1‘
         then echo Errore; exit 2;
       fi
       for i in $*  do
        case $i in
                  [0-9]) echo primo argomento $i;;
                   ?   ) echo carattere $i;;
                   *    ) echo Non un carattere; exit 3;;
            esac
      done;;
esac
# controllo ben fatto?
```

```
primoarg=$1
shift

while read linea
do
    cont=0
    for i in $*
    do
        case $linea in
        *$i*) cont=‘expr $cont + 1‘;;
        esac
    done
    if test $cont -ge $primoarg
        then echo $linea
     fi
done
```

```
filtro 3 a b c A B C < inFile

filtro 3 a b c A B C < inFile | wc –l

cat inFile | filtro 3 a b c A B C | filtro 2 x y z X Y Z
```


FATTORIALE

Si noti l’uso della valutazione in backquote

 # fatt argnumero

```
#SOLUZIONE ITERATIVA
case $# in
    1) ;;
    *) exit 1;;
esac
fatt=1 ; x=1;
while test $x -le $1;  do
    fatt=‘expr $fatt \* $x‘;    x=‘expr $x + 1‘
done
echo fatt vale $fatt
```

 # fatt argnumero

```
# SOLUZIONE RICORSIVA
case $# in
    1) ;;
    *) exit 1;;
esac
if test $1 -le 1 ;   then  fatt=1
else
            xmeno1=‘expr $1 - 1‘
            temp=‘fatt $xmeno1‘
            fatt=‘expr $temp \* $1‘
fi
echo $fatt
```

FILE COMANDI
STRUTTURA DI UN FILE COMANDI

SENZA SCANSIONE DI UNA GERARCHIA DIRETTORI
Controllo argomenti
Algoritmo

CON SCANSIONE DI GERARCHIA DI DIRETTORI
Controllo argomenti
Modifica della variabile PATH

Algoritmo	Ricorsione
Ricorsione	Algoritmo

MODIFICA della VARIABILE PATH

PATH=/**home/antonio/bin**:$PATH

/**home/antonio/bin** è il direttorio che contiene il file comandi da invocare durante la ricorsione
Si devono usare nomi assoluti e validi per ogni possibile utente invocante (ad esempio non si deve usare $HOME, pwd, o ogni altro riferimento vincolato ad un utente o processo specifico)

In altre parole, NON si devono usare per la modifica nomi relativi, ma solo cammini assoluti validi per ogni possibile invocazione ed invocante

RICORSIONE o meglio RIENTRANZA

Ogni **file comandi** viene attivato una volta per ogni direttorio
e si crea un **processo diverso** che esplora il solo direttorio obiettivo

molte strutture possibili

RICORSIONE per SCANDIRE la GERARCHIA

```
for i in * ; do
  if test -d $i ; then
     invocazione file comandi con argomenti
  fi
done
```

Sono possibili due schemi principali per designare il direttorio obiettivo

A) l'invocante cambia direttorio

```
for i in * ; do
  if test -d $i ; then
     cd $i
# se $i e' un direttorio, lo shell corrente cambia direttorio
     invocazione file comandi con argomenti
# ritorna al direttorio padre precedente
     cd ..
  fi
done
```

A) l'invocato cambia direttorio

```
# se $2 e' il direttorio, lo shell lo fa diventare corrente
cd $i
```

come sopra

FILE COMANDI

la parte di controllo degli argomenti è fondamentale per qualunque programma

è necessario verificare che gli **argomenti siano corretti**

- **Prima nel numero giusto**
- **Poi del tipo richiesto**

#invocazione di comando per ...

```
case $# in
0|1|2|3|4)    echo Errore. Almeno 4 argomenti ... >&2
              exit 1;;
esac

# solo in caso di argomenti corretti: non possiamo usare $1
# se non siamo sicuri che esista
# primo argomento direttorio (o file)
     if test ! -d $1
     then echo argomento sbagliato: $1 direttorio >&2; exit 2
     fi;;

# primo argomento nome file assoluto
case $1 in
   /*) if test ! -f $1
       then echo argomento sbagliato: $1 file >&2; exit 2;fi;;
   * ) echo argomento sbagliato: $1 assoluto >&2; exit 3;;
 esac

# primo argomento nome solo relativo (stretto) al direttorio
case $1 in
*/*) echo argomento sbagliato: $1 nome relativo; exit 3;;
* ) ;;
esac
```



```
# secondo argomento stringa numerica
# MODO 0
# si potrebbe usare un case con match  [0-9]                          ?
# e case con match  [0-9] | [0-9][0-9]                                ?
# e case con match  [0-9] | [0-9][0-9] | [0-9][0-9][0-9]              ?

# MODO 1      il comando cut che consente di selezionare
# le colonne delle righe di un file
# si esegua in un ciclo per estrarre una lettera alla volta
# si metta in pipe l'argomento:  echo $2 | cut -c$num
# il singolo carattere può essere estratto ed esaminato
# num deve essere incrementato ad ogni ciclo


# MODO 2
# ogni carattere di un valore numerico deve essere numerico
# un solo carattere non numerico rende sbagliato l'argomento
case  $2 in
*[!0-9]* ) echo errore in argomento numerico; exit 4;;
*)  ;; # tutti i caratteri sono numerici
esac

# MODO 3
# il comando expr può verificare una espressione
# numerica (è necessaria la operazione)
# expr restituisce lo stato 0 in caso di successo
# valore 1 per risultato 0; 2 in caso di insuccesso
expr $2 + 0 >  /dev/null 2> /dev/null
if  test $? -ne 0
     then echo errore in argomento numerico: $i  >&2; exit 4
fi
# qual è la ragione delle ridirezioni su /dev/null?
```

Si possono eliminare alcuni argomenti per comodità di scansione: si usi lo shift, dopo avere salvato gli argomenti che vengono eliminati

```
salva1=$1  # salvataggio di $1
shift
salva2=$1  # salvataggio di $2
shift
# cosa vale adesso $*?
for i;  do
# il ciclo è fatto per tutti gli argomenti esclusi quelli tolti
done
# gli argomenti iniziali sono dati da $salva1 $salva2 $*
```

Per ottenere di trovare comandi:

- uso di **nomi assoluti**
- uso di **nomi relativi** (ampliando il PATH con i direttori in cui cercare)
  se il PATH è esportato, lo si fa solo nel primo shell

Si noti che un **file comandi**

- può **passare** a nuove invocazioni (file comandi o filtri eseguibili) un numero **qualunque di argomenti**
- può **passare** a nuove invocazioni un insieme di valori di **variabili esportate**
  copiate e non condivise successivamente
- non può **ottenere** alcun risultato di ritorno (a parte lo **stato** di ritorno intero)
- non può vedere qualunque modifica di variabili attuata da un file comandi (o filtro eseguibile) **invocato**
  se non registrata in un file del file system

Si veda l'uso di backquote, per condividere output