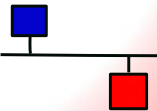


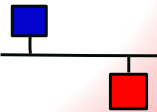
La Shell Lavora Per Noi - 2

Come accelerare compiti comuni
Utilizzando la Shell, la sua programmazione ed i
comandi di Linux
Clonazione schermo – ricerca risoluzioni comuni



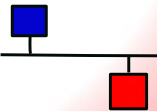
Dove siamo e dove andiamo

- Nella parte precedente abbiamo visto lo script che ci consente di clonare, metter a destra o mettere a sinistra il secondo schermo.
- Nella clonazione abbiamo ipotizzato che la risoluzione dei due schermi fosse identica
- Se così non fosse, la clonazione non avrebbe successo.
- Cerchiamo ora quindi di trovare una risoluzione comune



Soluzione adottata

- Per prima cosa devo **distinguere** tra la gestione di **clonazione** e di **destra e sinistra**.
- Ovviamente lo farò con un **if**
- In secondo luogo, dato che la gestione della clonazione è completamente diversa, decido di spostarla in una **funzione** che riceverà in ingresso le informazioni di **xrandr**
- Chiameremo questa funzione **setClone** e le passeremo come parametri il nome dello schermo **primario (\$PRIMARY)** e **secondario (\$SECONDARY)**



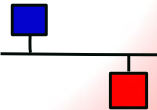
Funzioni

- Vediamo ora come è possibile creare delle **funzioni** che sono viste come comandi, con i loro parametri ed il loro valore di ritorno

- La definizione è molto semplice:

```
function <nome> {  
                                <comandi>  
}
```

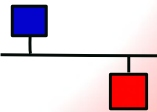
- <nome> è il nome con cui potremo richiamarla
- Ovviamente <comandi> è il corpo della funzione, con i suoi **for** ed **if** dove servono
- Se quando la richiamiamo dopo il nome mettiamo dei valori, questi potranno essere recuperati all'interno della funzione con le variabili **\$1**, **\$2** ...
- Se si vuole ritornare un valore (per esempio per usarla in un test) dovremo usare il comando **return** al posto del comando **exit**



Modifichiamo lo script originale

- Dovremo modificare lo script originale per:
 - Chiamare direttamente **xrandr** se la posizione è **left-of** o **right-of**
 - Richiamare la nostra funzione **setClone** passandole in ingresso il risultato di **xrandr** se la posizione è **same-as**
- Modificheremo quindi l'ultima riga (quella che richiama **xrandr**) così:

```
if [ $POSITION = 'same-as' ]
then
    xrandr | setClone $PRIMARY $SECONDARY
else
    xrandr --output $SECONDARY --$POSITION $PRIMARY
fi
```

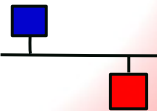


Impostiamo la nostra funzione

- Cominciamo a 'riempire' la nostra funzione
- Scriviamo la struttura della funzione ed assegniamo i parametri a due variabili con nomi significativi
- Impostiamo anche la lettura dell'input (il risultato di **xrandr**).
- Per questo utilizzeremo la funzione **read** con il parametro **-r** per non interpretare i \ nell'ingresso, all'interno di un **while**;

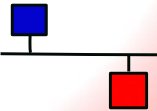
```
function setClone {  
    PRIMARY=$1  
    SECONDARY=$2  
    while read -r l  
    do  
        ...  
    done  
}
```

- Questo comando metterà ogni linea letta nella variabile **l** (in **\$l**)



I dati d'ingresso e la tecnica

- **Xrandr** produce ed invia all'ingresso della nostra funzione due tipi di linea:
 - **Descrizione** di display, che comincia con il nome del display
 - **Modeline**, che comincia con la risoluzione indentata di qualche spazio, seguita dai dati per programmare la scheda video
- Di fatto abbiamo un elenco di descrizioni di display seguita dall'elenco delle sue modeline
- A noi interessano tutte le possibili risoluzioni per ogni display
- Vogliamo produrre **una variabile per ogni display**, il cui nome sia derivato dal **nome del display** ed il cui contenuto sia la **lista delle risoluzioni** disponibili, separate da spazio
- Ad esempio per **eDP-1**:
eDP_1_MODES = 2560x1600 2560x1440 2048x1536 1920x1440 ...
- Dobbiamo sostituire i – che non possono stare nei nomi di variabile con degli _ e, più per estetica che per altro, aggiungiamo un **_MODES** in fondo al nome

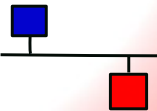


Estrazione del nome

- All'interno del ciclo riconosceremo i tipi di riga e li gestiremo
- Vediamo ora come si gestisce la **riga del nome**

```
if echo $l | grep -Eq "^[a-zA-Z][a-zA-Z0-9-]"
then
    DEV=`echo $l | sed -e "s/ .*$//"`
    MODELIST=`echo ${DEV} | sed -e "s/-/_/g"`_MODES
    typeset -n CML=$MODELIST
elif ... (segue riconoscimento della modeline)
```

- Per prima cosa, abbiamo la nostra riga nella variabile **l**, ma per riconoscerla dovremo usare **grep** che richiede il testo all'ingresso, quindi useremo **echo** per mettere in uscita **l** e la pipe per mandarla su **grep**
- Le righe con i dispositivi cominciano con una lettera, seguita da una lettera o un numero o un - (dobbiamo eliminare le righe che cominciano cose come **h:** e **v:**)
- Metteremo poi nella variabile **DEV** il nome del monitor, come abbiamo già visto prima
- Dobbiamo poi sostituire in **DEV** i - con i _ e metteremo il nome della lista così ottenuto in **MODELIST**
- Per finire creeremo, con **typeset -n** un link (riferimento) tra la variabile il cui nome contenuto in **MODELIST** ed il nome **CML**, per riferirci alla lista corrente con un nome standard

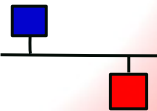


Prepariamoci per la ricerca

- Prima di cercare i modi comuni dobbiamo preparare alcune variabili e collegamenti:

```
PRIMARY_LIST=`echo ${PRIMARY} | sed -e "s/-/_/g"`_MODES
SECONDARY_LIST=`echo ${SECONDARY} | sed -e "s/-/_/g"`_MODES
typeset -n PRIMARY_MODES=$PRIMARY_LIST
typeset -n SECONDARY_MODES=$SECONDARY_LIST
MODE=""
```

- Elaboriamo i nomi dei monitor **PRIMARY** e **SECONDARY** come prima per produrre i relativi nomi delle liste in **PRIMARY_LIST** e **SECONDARY_LIST**
- Utilizziamo ancora **typeset -n** per creare due link alle liste reali preparate prima
- Prepariamo una variabile **MODE** vuota che conterrà il modo comune, se verrà trovato.

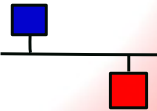


Aggiunta dei modi

- Ora, alla variabile appena creata e alla quale ci possiamo riferire tramite il link **CML** dobbiamo associare i modi, che cominciano sempre con una cifra.
- Completiamo l'elif della slide precedente:

```
elif echo $l | grep -q "^[0-9]"
then
    MODE=`echo $l | sed -e "s/ .*$//"`
    CML="$CML $MODE"
fi
```

- Usiamo **echo** e **grep** come prima, per cercare righe che comincino con una cifra
- Usiamo il solito procedimento per mettere il primo pezzo – la risoluzione – nella variabile **MODE**
- Alla fine aggiungiamo il nuovo modo alla variabile **CML**, che è un riferimento alla lista relativa al nostro monitor.

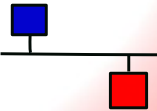


Cerchiamo il modo comune

- Ora dobbiamo trovare un modo comune.
- Confronteremo uno ad uno i modi della lista del primario con quelli del secondario fino a che non ne troveremo uno uguale:

```
for pm in $PRIMARY_MODES
do
    for sm in $SECONDARY_MODES
    do
        if [ $pm = $sm -a -z $MODE ]
        then
            MODE=$pm
        fi
    done
done
```

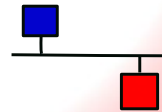
- Creiamo un doppio ciclo annidato dove, per ogni modo del primario **pm**, lo confrontiamo con ogni modo del secondario **sm** e, se sono uguali e la variabile **MODE** è vuota, assegneremo il modo trovato alla variabile.
- Verifichiamo che **MODE** sia per catturare il primo tra i modi che è presente in entrambe le liste (quello con risoluzione maggiore)



Utilizziamo il risultato

- Ora non ci resta che utilizzare il risultato: cambiare la risoluzione dei due monitor a quella comune e dare il comando di clonazione, a condizione che sia stato trovato un modo comune:

```
if [ -z "$MODE" ]
then
    echo Nessun modo comune ERRORE!
    exit 2
else
    echo $MODE
    xrandr --output $PRIMARY --mode $MODE
    xrandr --output $SECONDARY --mode $MODE
    xrandr --output $SECONDARY --same-as $PRIMARY
fi
```



La funzione completa

```
function setClone {
    PRIMARY=$1
    SECONDARY=$2
    while read -r l
    do
        if echo $l | grep -Eq "^[a-zA-Z][a-zA-Z0-9-]"
        then
            DEV=`echo $l | sed -e "s/ .*$/"`
            MODELIST=`echo ${DEV} | sed -e "s/-/_/g"`_MODES
            typeset -n CML=$MODELIST
        elif echo $l | grep -q "^[0-9]"
        then
            MODE=`echo $l | sed -e "s/ .*$/"`
            CML="$CML $MODE"
        fi
    done
    PRIMARY_LIST=`echo ${PRIMARY} | sed -e "s/-/_/g"`_MODES
    SECONDARY_LIST=`echo ${SECONDARY} | sed -e
"s/-/_/g"`_MODES
    typeset -n PRIMARY_MODES=$PRIMARY_LIST
    typeset -n SECONDARY_MODES=$SECONDARY_LIST
    MODE=""
```

```
for pm in $PRIMARY_MODES
do
    for sm in $SECONDARY_MODES
    do
        if [ $pm = $sm -a -z $MODE ]
        then
            MODE=$pm
        fi
    done
done
if [ -z "$MODE" ]
then
    echo Nessun modo comune ERRORE!
    exit 2
else
    echo $MODE
    xrandr --output $PRIMARY --mode $MODE
    xrandr --output $SECONDARY --mode $MODE
    xrandr --output $SECONDARY --same-as $PRIMARY
fi
}
```

