Ambienti Operativi

SUPSI Dipartimento Tecnologie Innovative

Gianni Grasso

4 ottobre 2024

Classe: I1B

Anno scolastico: 2024/2025

Indice

1 Introduzione

Per riuscire a capire cos'è un ambiente operativo, dobbiamo innanzitutto riuscire a vedere il computer come uno strumento di elaborazione dei dati alla quale vengono passati degli input e date delle istruzioni. L'ambiente operativo è il luogo nella quale vengono date queste istruzioni.

È importante non fare confusione e non confondere il sistema operativo con l'ambiente operativo, il primo ha lo scopo di nascondere i meccanismi di gestione della macchina, rendendo l'utente in grado di poterla utilizzare senza conoscerne il funzionamento a basso livello, l'ambiente operativo invece fa da tramite tra gli utenti ed il sistema operativo e può essere visto come l'interfaccia nella quale si danno istruzioni alla macchina.

1.1 Tipi di file

Tutti i dati che si trovano su un computer sono rappresentati da una sequenza binaria, con **file binari** intendiamo che i dati non sono direttamente comprensibili da una persona mentre per con il termine **file personali** si intendono i file che possono essere compresi da una persona sotto forma di testo.

Per codificare i dati testuali si associa ogni carattere a una sequenza binaria, non esiste però un'unica codifica dei caratteri, di seguito sono riportati alcuni esempi:

- ASCII
- Windows code pages
- ISO 8859
- Unicode
- ...

Dobbiamo poi fare distinzione tra documenti testuali semplici e documenti strutturati, i primi sono quei documenti in cui la struttura logica non è facilmente distinguibile mentre nel secondo caso parliamo di documenti di testo in cui c'è una struttura che stabilisce il contenuto del file (ad esempio i file **csv**.

1.2 SSH

SSH, ovvero Secure Shell, è un protocollo di rete crittografico che ci consente di utilizzare servizi di rete in modo sicuro su una rete non protetta. Le sue applicazioni più comuni sono il login remoto e l'esecuzione da riga di comando, noi useremo il SSH per connetterci ad un server didattico.

Per connetterci al server da macOS/Linux dobbiamo digitare a terminale il seguente comando:

```
ssh linux1-didattica.supsi.ch -l nome.cognome@supsi.ch
```

1.3 Struttura di un filesystem

Il filesystem di un sistema ha una struttura ad albero, la radice è la cartella **root (/)**, tutte le altre directory sono sottostanti ad essa.

Tra le directory principali che ci interessano ci sono **/home**, cartella che contiene le directory degli utenti locali, e **/tmp** cartella nella quale risiedono i file temporanei.

Un percorso può essere assoluto o relativo, nel primo caso specifichiamo l'intero percorso di una directory o un file, indipendentemente da dove ci troviamo, mentre nel secondo indichiamo il percorso per raggiungere un file a partire dalla posizione corrente.

2 Bash - Introduzione

La prima volta che apriremo un terminale potremo notare che il cursore è preceduto da: **utente@host:path$**, dove utente sta per il nome dell'utente connesso alla macchina, host il nome del server e path il percorso corrente.

È importante ricordare che in bash è tutto **case-sensitive**, sia i comandi che i nomi dei file e delle directory.

2.1 Comandi

La struttura di un comando da terminale è definita in questo modo:

comando [−OPTION] ... [ARGUMENT] ...

Dove OPTION indica le opzioni (-a, -b, ...) e ARGUMENT gli argomenti, che possono essere obbligatori oppure no a dipendenza del comando.

Per consultare il manuale di un comando digitare:

man comando

2.2 Globbin

Il Globbin consiste nell'utilizzare un pattern con uno o più caratteri "wildcard" per trovare o fare azioni sui file.

- *, corrispondenza con zero o più caratteri qualsiasi
- ?, corrispondenza con esattamente un carattere
- [ ], corrispondenza tra un gruppo di caratteri
- [a-z], un carattere dalla 'a' alla 'z'
- [^abc], non-corrispondenza tra un gruppo di caratteri

3 Redirezione

Sui sistemi Unix i programmi hanno accesso a tre stream di input e output.

- **stdin** (standard input) [0], input dalla tastiera
- **stdout** (standard output) [1], output su schermo
- **stderr** (standard error) [2], output degli errori su schermo

Possiamo redirezionare gli input e output di default. Per redirezionare l'input useremo il carattere '<', per l'output '**1>**' (sovrascrive) o '**1>>**' (append) e per gli errori '**2>**'. Ad esempio:

```
utente@host:~$ cat leggi.txt >> aggiungi.txt
```

Redireziona lo stdout del comando *cat* sul file *aggiungi.txt*.

Per redirezionare sia lo stdout che lo stderr si usa '**>&**'. Inoltre è possibile redirezionare un qualsiasi output a **/dev/null** per non visualizzarlo.

3.1 Pipe

La pipe è un buffer che permette di passare lo stdout di un comando come stdin di un altro, da sinistra a destra. Il seguente comando ad esempio prende l'input del comando *cat* e lo passa al comando *grep*:

```
utente@host:~$ cat A.txt| grep "ABC"
```

È possibile redirezionare lo stderr nello stdout, in questo caso possiamo usare '**2>&1**':

```
utente@host:~$ ls nonesistente 2>&1 >/dev/null | grep impossibile
```

Ecco una lista di comandi utili:

- head, restituisce le prime n rghe di un file
- tail, restituisce le ultime n rghe di un file
- tr, sostituisce una lettera con un altra all'interno di un file
- cut, estrae i primi n caratteri di ogni linea di un file
- grep, visualizza tutto ciò che contiene una determinata stringa
- sort, ordina il contenuto dell'output, di default in ordine alfabetico
- wc, conta il numero di righe di un output
- uniq, stampa a schermo le righe di un output senza duplicati (righe adiacenti uguali)

4 Esecuzione sequenziale e condizionale

È possibile eseguire sequenzialmente una serie di comandi inserendoli in un unica linea di comando, ogni istruzione deve essere separata da un **;**. Ad esempio:

```
utente@host:~$ echo "ciao"; echo "mondo"; pwd
ciao
mondo
/home/utente
```

È anche possibile eseguire più comandi inserendoli sulla stessa linea di comando legandoli insieme con un operazione booleana, **OR** || oppure **AND** &&. Ad esempio:

```
utente@host:~$ cat miofile || date || ls
```

L'eseuzione termina quando uno dei comandi (eseguiti da sinistra verso destra) termina correttamente senza errori.

```
utente@host:~$ cat miofile && date && ls
```

L'esecuzione termina quando uno dei comandi (eseguiti da sinistra verso destra) produce un errore.