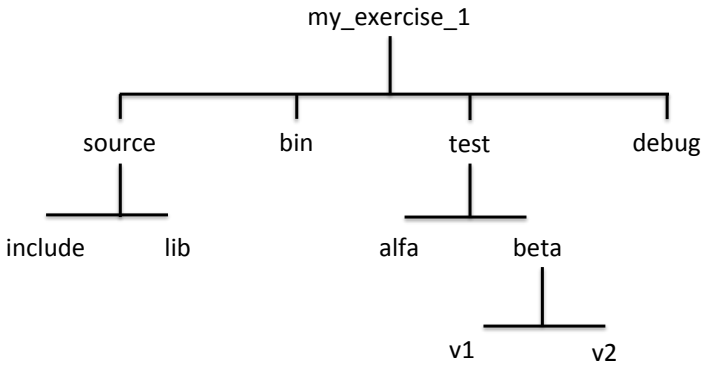
Sistemi Operativi – Lab 1 14.10.19/15.10.19 - A.A. 2019/2020 - Prof. L. Sterpone

Goal: Gestire i file e i direttori tramite i comandi UNIX/Linux. Familiarizzare con gli editor di testo..

Esercizio 1 (gestione file e direttori): Creare il seguente albero di directory nella propria home utilizzando i comandi UNIX/Linux.



Esercizio 2 (editor generico): Attraverso l'uso di un editor a scelta tra quelli forniti dal sistema operativo creare un file di testo denominato edito.txt a piacere costituito da 5 righe e salvarlo all'interno del direttorio my_exercise_1/source. Eseguire quindi le seguenti operazioni: 1) copiarlo all'interno del direttorio alfa e beta usando path assoluti e path relativi. Verificare l'esistenza del file nei rispettivi direttori attraverso i comandi more, less e cat. 2) editare il file edito.txt nel direttorio alfa attraverso VI: inserire una riga di testo nel file. Quali comandi sono necessari ?

Esercizio 3 (copia e rimozione): Utilizzare i comandi mkdir e cp per copiare il medesimo albero di direttori my_exercise_1 denominandolo my_copy_1 e my_copy_2. Utilizzare per my_copy_1 il metodo ricorsivo. Rimuovere quindi l'albero my_copy_1. Quale vantaggio ottengo utilizzando i cp e rm in modo ricorsivo ?

Esercizio 4 (gestione permessi): Modificare i diritti di accesso al direttorio my_copy_2 eliminando tutti i possibili accessi. Posso ancora accedere al direttorio ? E' possibile accedere al direttorio senza modificare nuovamente i diritti di accesso ?

Esercizio 5 (compilazione): Scrivere un programma in linguaggio C che utilizzando le primitive I/O ANSI C (fgetc/fputc, fscanf/fprintf, fget/fputs a scelta) sia in grado di copiare un file di testo in un file identico. Creare il file in modo tale che i nomi dei due file (sorgente e copia destinazione) siano gestiti tramite linea di comando. Effettuare la compilazione tramite gcc.

Esercizio 6 (approfondimento): E' possibile creare il direttorio dell'esercizio 3 (my_copy_1) evitando di replicare i contenuti dei file ovvero non utilizzando il comando di copia. In quale modo ?

Sistemi Operativi – Lab 2 21.10.19 - A.A. 2019/2020 - Prof. L. Sterpone

Goal: Approfondimento sul file system.

Esercizio 1 (gestione file, direttori): Scrivere un programma in linguaggio C che utilizzando le primitive I/O ANSI C (fgetc/fputc, fscanf/fprintf, fget/fputs a scelta) sia in grado di copiare un file di testo in un file identico. Creare il file in modo tale che i nomi dei due file (sorgente e copia destinazione) siano gestiti tramite linea di comando. Effettuare la compilazione tramite gcc.

Esercizio 2 (comandi generici linux):

1. All'interno della vostra home directory create 2 directory denominate “first” e “second”.
2. Copiate il file /etc/profile nella directory uno, conservandone il nome.
3. Copiate il file /etc/profile nella directory due cambiandone il nome in copia-profile.
4. Spostate il file profile nella directory due ed il file copia-profile nella directory uno.
5. Cancellate i due file con uno stesso comando.
6. Cancellate le due directory vuote.
7. Verificate il funzionamento del comando *touch* per la modifica dell'orario di ultimo accesso e per la creazione di un file vuoto

Sistemi Operativi – Lab 3 28.10.19 - A.A. 2019/2020 - Prof. L. Sterpone

Goal: Approfondimento sul file system.

Esercizio 1 (gestione file, direttori, makefile): si scriva un programma in linguaggio C in grado di ricevere sulla riga di comando il path assoluto di un direttorio e ne visualizzi il contenuto.

Si compili il programma mediante gcc e ne si verifichi il funzionamento confrontandone i risultati con il comando "ls".

Compilare il programma mediante un Makefile contenente lo specifico target di compilazione.

Nota: Utilizzare le corrette opzioni GCC in modo tale da dividere compilazione e linking per generare opportunamente il file oggetto .o.

Esercizio 2 (comandi generici linux):

Esplorare l'esecuzione dei seguenti comandi (si usi il comando man per visionare le proprietà d'uso, esempio man cat) e se ne deduca il funzionamento:

1. cat e more nome_file
2. df
3. touch nome_file
4. ln –s /m_dir mio_direttorio (m_dir è un direttorio preliminarmente creato dall'utente).
5. cat nome_file | sort
6. cat < nome_file| sort
7. alias lista='ls –l'
8. sort < nome_file
9. ps
10. ps f

Sistemi Operativi – Lab 4 04.11.19 - A.A. 2019/2020 - Prof. L. Sterpone

Goal: Approfondimento sul file system ed introduzione ai processi

Esercizio 1 (gestione file, direttori, makefile): Si scriva un programma in linguaggio C in grado di ricevere sulla riga di comando il path assoluto di un direttorio e ne visualizzi "ricorsivamente" il contenuto.

Si compili il programma mediante gcc e ne si verifichi il funzionamento confrontandone i risultati con il comando "ls –R".

Compilare il programma mediante un Makefile contenente lo specifico target di compilazione.

Modificare il Makefile aggiundendo il target "clean" che rimuove il file oggetto nel direttorio corrente.

Nota: Utilizzare le corrette opzioni GCC in modo tale da dividere compilazione e linking per generare opportunamente il file oggetto .o.

Esercizio 2 (esempio di generazione processo figlio):

Si considerino i seguenti programmi di esempio (a, b). Si includano le opportune librerie per la gestione dei processi. Si scrivano e si compilino i programmi verificandone successivamente l'esecuzione con e senza le differenti system sleep (esempio numero 1) e descrivere attraverso AGP e CFG il comportamento dell'esempio numero 2.

a)

```
int main(void)
{
        printf("A\n");
        int pid = fork();
        printf("B\n");
        if (pid)
           {
                Printf("Father waiting\n");
                wait(NULL);
        }
    else
          {
           printf("C\n");
           //sleep(5)
           exit(0);
           }
}
```

b)

```
int main(void)
{
        int x = 0;
        if (fork() == 0)
        {
                x++;
                if (fork() == 0)
                        x++;
                else
                        wait(NULL);
        }
        else wait(NULL);
        printf("%d\n", x);}
```