Università di Verona

Corso di Laurea in Informatica/Informatica Multimediale
Laboratorio di Sistemi Operativi
26 Settembre 2008

Esercizio 1 (16 punti)
Il comando UNIX "find" effettua la ricerca di una stringa di caratteri (*string*) negli alberi di directory aventi radice in ognuno dei nomi di file specificati da un percorso (*path*). Sintassi:

find `opzioni` {`path`} `-name` {string}

Scrivere uno script bash che si basi sul comando **find** per cercare tutti i file contenenti una determinata stringa, partendo da una cartella precisa del file system. Lo script deve ricevere 4 argomenti nella linea di comando:

- *string*, ovvero la stringa da ricercare.
- *path*, ovvero il percorso completo da "/" della cartella da cui fare la ricerca.
- Il *file_output.log*, ovvero il percorso completo dove salvare tutto ció che il comando manda sullo standard **output**.
- Il *file_error.log*, ovvero il percorso completo dove salvare tutto ció che il comando manda sullo standard **error**.

Lo script deve controllare che il numero di argomenti della linea di comando sia giusto ed in caso contrario deve visualizzare un messaggio indicante il corretto utilizzo dello script. Infine lo script deve visualizzare un messaggio di errore anche nel caso in cui la directory fornita come argomento alla linea di comando non esista.

Esercizio 2 (16 punti)
Scrivere un programma C per la gestione della ricerca di stringhe tramite `find`.
Si crei un processo padre P che deve richiedere all'utente: la stringa di cui fare la ricerca (*string*) e la directory (*path*) da cui partire con la ricerca della stringa.
Il processo P deve inoltre successivamente creare un secondo processo F (figlio) che esegue il comando `find`.
Il processo F deve eseguire la ricerca della stringa a partire dalla cartella specificata tramite *path* sullo standard output il risultato del comando di sistema */bin/find*.
Il processo P attesa la terminazione del figlio (e l'esecuzione del comando `find`) termina.

Eventuali System Call da utilizzare:

```
#include <stdio.h>                          #include <unistd.h>
#include <sys/wait.h>                       #include <sys/types.h>
int pipe(int filedes[2]);          pid_t fork(void);          int kill(pid_t pid, int sig);
int execle(const char *path, const char *arg , ..., char * const envp[]);
int execl(const char *path, const char *arg, ...);
int execlp(const char *file, const char *arg, ...);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
int dup2(int oldfd, int newfd);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
int close(int fd);                          int dup(int oldfd);
int open(const char *pathname, int flags, mode_t mode);
int creat(const char *pathname, mode_t mode);
sighandler_t signal(int signum, sighandler_t handler);
pid_t wait(int *status);           pid_t waitpid(pid_t pid, int *status, int options);
```