Documentazione Xlogger

Usage

`./bin/xlogger [OPTION]... "[SCRIPT]"`

Informazioni di compilazione e esecuzione a fine documento.

Features

Features notevoli che abbiamo implementato:

- **Logger**

  Il processo che si occupa di scrivere le statistiche raccolte su file (logger) è implementato tramite demone custom, cioè non si utilizza una libreria per crearlo. Inoltre, il logger è univoco e comune a qualsiasi istanza del tool xlogger, pertanto se su uno stesso sistema sono in esecuzione più istanze del tool, esse comunicheranno le statistiche raccolte al logger che scriverà su file.

  Il logger stesso viene lanciato all'esecuzione della prima istanza del tool e chiuso al termine dell'ultima istanza, evitando di rimanere in background alla chiusura di tutte le istanze.

- **Operatori**

  Vengono gestiti i seguenti operatori: `&&`, `||`, `;`, `|`, `<`, `<<` e infine `>`. I primi quattro sono operatori utilizzabili tra un sottocomando e l'altro, gli ultimi tre possono specificare ridirezioni dell'output su file per i singoli sottocomandi. Sono tutti utilizzabili più volte nello stesso comando, anche mescolati tra di loro. Gli operatori logici (`&&`, `||` e `;`) sono valutati in maniera "lazy", aspettando l'esecuzione del sottocomando di sinistra prima di eseguire il sottocomando di destra. Il comportamento della pipe è specificato di seguito.

- **Esecuzione e pipe**

  Il processo principale genera un processo per ogni sottocomando, che viene eseguito tramite `execvp`. In seguito, le statistiche vengono raccolte dal processo principale, che pertanto aspetta la fine dell'esecuzione di ogni sottocomando. In caso di operatore `|`, il processo principale utilizzerà dei thread per raccogliere le statistiche: si avrà quindi la continua esecuzione dei sottocomandi in maniera non bloccante.

- **Gestione di sottocomandi**

  Se lo `SCRIPT` contiene sottocomandi (per esempio `ls | wc` o `echo 1; ls`), questi vengono eseguiti separatamente, con statistiche raccolte in modo individuale e raggruppate alla stampa a seconda del comando di appartenenza.

- `cd`

  Viene gestito il sottocomando `cd`, che effettua il cambio della directory corrente per tutti i prossimi sottocomandi. Essendo comunque un sottocomando valido, vengono raccolte e loggate le sue statistiche.

- **Statistiche**

  Raccolta di una serie di statistiche, visualizzabili in un normale file di log. La raccolta avviene tramite il riempimento di una struttura di libreria `rusage`, fornendo statistiche dettagliate come da esempio:

```
############################## STATS ############################
#
#      2018-05-27 21:32:40
#
#      whole command: pacman -Syu
#      n° subcommand: 1
#           username: root (0)
#
#    ========================================================
#
#                subcommand: pacman -Syu
#
#                       PID:         14374
#                      PGRP:         14371
#                       SID:          3924
#               exit status:             0
#              elapsed time:      0.758215 s
#             CPU time used:      0.016561 s
#              max ram size:         27612 kB
#           soft page faults:          5287
#           hard page faults:             0
#                      swaps:             0
#           signals received:             0
#      vol. context switches:            13
#      inv. context switches:            17
#
#################################################################
```

Compilazione

È necessario eseguire il tool `make` nella cartella del progetto. A partire da i file `.c`, verranno creati i file oggetto salvati nella cartella temporanea `/bin` e linkati a creare l'eseguibile `xlogger`.

La regola `make clean` rimuove la directory `/bin` e eventualmente la cartella del tool creata nella directory `/tmp` nella root.

Esecuzione

L'eseguibile `xlogger` nella cartella `/bin` si occupa di eseguire un programma `SCRIPT` definito dall'utente e di raccogliere statistiche riguardanti il programma stesso. Le statistiche vengono mandate a un processo "logger", implementato come un demone, che scrive la statistiche su file.

Vi sono una varietà di flag passabili al tool per modificare il suo comportamento:

- `-a`, `-appout=FILE`: permette di appendere l'output dello `SCRIPT` ad un `FILE` a discrezione dell'utente;
- `-A`, `-apperr=FILE`: identico al flag precedente, eccetto che appende gli errori invece che l'output;
- `-e`, `--printerr`: stampa gli errori a video, comportamento di default;
- `-f`, `--format=FORMAT`: specifica il formato del file di log contenente le statistiche. Può essere `"txt"` (default) o `"csv"`;
- `--nerr` e/o `--nout`: non stampa errori o output dei sottocomandi, rispettivamente. Stampa eventuali errori o output del tool normalmente;
- `-o`, `--printout`: stampa l'output dei sottocomandi a video;
- `-p`, `--path=FILE`: specifica il file dove scrivere le statistiche;
- `-q`, `--quiet`: non stampare assolutamente niente a video, nemmeno errori o output del tool;
- `-s`, `--show`: stampa le statistiche contenute nel file di log direttamente a video nel terminal;
- `-s`, `--showcust=FILE`: stampa le statistiche contenute nel file di log personalizzato direttamente a video;
- `-x`, `--overout=FILE`: sovrascrivi il FILE durante la stampa degli output dei sottocomandi a quel FILE;
- `-X`, `--overerr=FILE`: sovrascrivi il FILE durante la stampa degli errori dei sottocomandi a quel FILE;
- `--help`: mostra la schermata di help e termina;
- `--version`: stampa la versione corrente del tool e termina.