Utilizarea Sistemelor de Operare: Tema 3

Termen de predare: **11 ianuarie 2012, ora 23:55**

Cuprins

Introducere

Se cere să scrieți un script Bash care să genereze statistici pe baza unui repository de cod sursă. Repository-ul este în format Git. Această temă este o bună ocazie să exersați shell scripting pentru testul practic și pentru întrebările care pot apărea din cursurile de shell scripting.

Revision control, numit și *version control* sau *source control* este procedura prin care codul sursă al unui program este stocat astfel încât starea sa la un anumit moment de timp să poată fi recuperată. Unitatea de bază cu care lucrează un program de revision control este *commit*-ul, o modificare unitară asupra unui fișier sau set de fișiere. Exemple de commit-uri sunt adăugarea unui feature, rezolvarea unui bug, sau chiar corectarea unei greșeli gramaticale. Pentru mai multe detalii despre revision control, folosiți Wikipedia: `http://en.wikipedia.org/wiki/Revision_control`.

Un sistem de versionare *centralizat* ține codul într-un loc central (numit, în general *repository*); pentru a schimba starea repository-ului, un programator trebuie să aibă acces de la liderul proiectului, în general sub forma unui user și parolă. Orice copie locală (*working copy*) a unui repository reflectă starea repository-ului la un moment dat; pentru a avea acces la istorie, se contactează repository-ul central.

La polul opus, există sisteme de versionare *distribuite*, la care se poate *clona* un repository împreună cu toată istoria sa. Mai mult, un programator poate crea modificări locale, fără a fi nevoie să le publice.

Pentru mai multe detalii despre sisteme de versionare distribuite și centralizate puteți consulta Wikipedia: `http://en.wikipedia.org/wiki/Distributed_revision_control`.

Git este un program care face revision control distribuit. Se remarcă prin viteza sa, dar are și o reputație de a fi dificil de înțeles la început. A fost dezvoltat de autorul kernel-ului Linux (Linus Torvalds) și este folosit de foarte multe proiecte open source. Există site-uri care găzduiesc gratuit repository-uri Git, cum ar fi `http://github.com/`.

Este important să vă familiarizați cu câteva concepte Git înainte de a porni rezolvarea temei.

Puteți folosi tutorial-ul de la `http://gitimmersion.com/`. De asemenea, este recomandat să citiți în detaliu *toată* tema înainte de a începe rezolvarea ei.

Pentru această temă va trebui să instalați Git (este disponibil ca pachet pe toate distribuțiile) pe un sistem Linux. Această temă *nu folosește o mașină virtuală*. Puteți folosi orice distribuție Linux pentru dezvoltare (inclusiv una în mașină virtuală). Procedura de testare este descrisă în secțiunea Testare.

Descrierea temei

Tema constă în scrierea unui script Bash care generează câteva statistici pe baza unui repository Git. Script-ul **trebuie** să se numească `stats`, acest nume este folosit de script-ul nostru de testare.

Apelare

Script-ul va putea fi apelat astfel:

```
./stats repository [-n maxcommits] operație [alte opțiuni]
```

- Parametrul `repository` este un director local care conține repository-ul pe care vor fi generate statisticile.
  - Practic, acest director este rezultatul unei comenzi `git clone`.
  - Nu este nevoie să faceți voi încă un clone, puteți lucra direct pe repository.
  - Este important, în schimb, să **nu modificați în niciun fel repository-ul**.
  - Orice operație, dacă nu e specificat altfel, trebuie desfășurată pe branch-ul `master`. Este responsabilitatea script-ului vostru să se asigure că acel branch este activ.
- Parametrul `-n maxcommits` specifică faptul că script-ul nu trebuie să analizeze toată istoria codului, ci doar ultimele `maxcommits` commit-uri.
  - Parametrul este opțional; în tot documentul vom marca parametrii opționali prin paranteze drepte.
  - Parametrul poate apărea doar între `repository` și `operație`, nu oriunde în linia de comandă. Folosiți această precizare pentru a vă simplifica munca.
- Parametrul `operație` descrie ce tip de statistică va genera script-ul. Valorile sale posibile sunt descrise în secțiunile următoare.
  - Unele operații pot avea unul sau mai mulți parametri; aceștia sunt descriși în secțiunea corespunzătoare operației.

Statistica top_contributors

Această statistică afișează toți autorii, sortați în ordine inversă după numărul de contribuții.

- Numele unui autor este **exact** cel dat de linia `Author:` a unui commit. Nu este nevoie să "normalizați" în vreun fel acest nume: nu eliminați adrese de mail, nu coagulați autori cu același nume dar adrese de mail diferite etc.
- În listarea finală, numele unui autor este prefixat de numărul de commit-uri pe care îl are.

- Autorii sunt listați în ordine inversă a numărului de contribuții.
- Parametrul opțional specifică afișarea doar a primilor N autori în ordinea numărului de commit-uri.

În secțiunea Testare există instrucțiuni despre cum puteți vedea exemple de rulare. Citiți întreaga temă înainte de a pune întrebări. Un *sample run* vă răspunde la multe întrebări.

Statistica added_lines

Această statistică afișează numărul de linii adăugate de commit-urile analizate.

- Reamintim că vi se poate cere să analizați toate commit-urile sau doar ultimele N (parametrul `-n`).
- Dacă opțiunea `added_lines` nu este urmată de un parametru, trebuie să faceți o statistică a liniilor adăugate *în toate fișierele*.
- Dacă opțiunea este urmată de un parametru, acest parametru este numele fișierului pentru care trebuie numărate liniile adăugate. Interpretați acest parametru ca o cale relativă la directorul repository-ului.

Statistica deleted_lines

Această statistică este similară cu precedenta, doar că numără liniile *șterse* dintr-un fișier (sau toate). Precizările secțiunii anterioare se aplică și aici.

Este recomandat să încercați să duplicați cât mai puțin cod: pentru că operațiile `added_lines` și `deleted_lines` sunt atât de similare, probabil pot folosi același cod în mare parte.

Statistica touched_files

Această statistică numără câte fișiere au fost "atinse" (modificate în vreun fel).

- Parametrul opțional al statisticii este un șir de caractere reprezentând căutarea după autor. Recomandarea noastră este să trimiteți exact acest șir ca argument la `git log --author`. Nu încercați să îl procesați, probabil veți obține rezultate diferite și veți pica testele.
- Dacă parametrul lipsește, se vor număra toate fișierele modificate în commit-urile selectate, de către toți autorii.
- Nu trebuie să numărați un fișier de două ori: dacă doi autori au modificat același fișier, el contează ca un singur fișier, nu ca două.

Statistica diff_branches

Această statistică listează commit-urile care nu sunt comune între două branch-uri.

- Evident, restricția despre funcționarea pe branch-ul master nu se aplică. Script-ul va lua în considerare cele două branch-uri care urmează ca parametri după numele operației.
- Commit-urile sunt listate în ordine invers cronologică și reprezentate prin "mesajul scurt" ce le descrie (prima linie din fiecare mesajul de commit).

- Pentru această statistică, parametrul `-n` nu are relevanță. Trebuie să listați toate commit-urile diferite între cele două branch-uri.
- Statistica nu are sens fără cei doi parametri care specifică branch-urile.

Testare

Testare locală

Pentru simplificarea procesului de corectare a temelor, dar și pentru a reduce greșelile temelor trimise, corectarea se va realiza automat cu ajutorul unor test publice.

Puteți descărca testele de la adresa `http://elf.cs.pub.ro/uso/res/uso-tema3-checker.tar.bz2`. Conținutul arhivei este următorul:

- `check` este script-ul care verifică tema voastră.
- `_checker` este un director ce conține repository-urile de test și cazurile de test.

Arhiva trebuie despachetată în directorul unde vă rezolvați tema. O dată despachetată, puteți face modificări scriptului de testare pentru a depana programul vostru, dar **testarea finală de pe vmchecker folosește arhiva originală**.

Nu este recomandat să modificați repositori-urile (disponibile în `_checker/repos`), dar, dacă le modificați, puteți să le recuperați pe cele originale din arhivă.

În `_checker/tests` sunt toate testele, după următoarea structură:

`_checker/tests/operație/repository` – acest director conține toate testele care corespund operației și repository-ului din cale. În el, găsiți unul sau mai multe fișiere de forma `01.*`. Semnificația lor este:

- `01.sh` este comanda rulată de test, efectiv operația pe care trebuie să o implementați corect pentru a trece testul. Sunt folosite variabilele `$stats` (tot timpul `./stats`, numele script-ului vostru) și `$repo`, repository-ul dat ca parametru, luat din calea pe disc a directorului.
- `01.ok` este output-ul corect al script-ului pentru comanda din `01.sh`. Voi trebuie să obțineți un output identic.
- `01.score` este punctajul pe care îl primiți dacă treceți testul; punctajul maxim este 100.
- Unele directoare conțin mai multe seturi de fișiere (`01`, `02`), semnificând faptul că există mai multe teste pentru același repository și operație.

Puteți consulta fișierul `.ok` corespunzător pentru a compara de mână ieșirea script-ului vostru cu cea oficială. **Nu are sens să modificați fișierele .ok**, copiile voastre locale nu vor fi trimise o dată cu tema.

Script-ul `check` poate fi apelat fără niciun parametru, caz în care rulează toate testele, sau în unul dintre cele două cazuri de mai jos:

- Cu un parametru, semnificând numele operației. De exemplu, `./check added_files` rulează toate testele corespunzătoare operației `added_files`.
- Cu doi parametri, semnificând numele operației și repository-ul.

Nicio parte din checker nu este "secretă". Puteți citi script-ul `check` dacă credeți că vă ajută și puteți folosi fișierele `.ok` pentru a vă clarifica formatul de output.

Nu are sens să trimiteți o temă care doar afișează conținutul fișierelor `.ok`, output-ul trebuie să fie generat folosind `git` și repository-ul dat ca parametru.

Trimiterea temei

Tema va fi trimisă pe vmchecker, încărcată sub forma unei arhive. Detaliile de conectare sunt aceleași:

- Adresa: `https://elf.cs.pub.ro/vmchecker/`
- User și parolă: aceleași de pe `cs.curs`.

Va trebui să trimiteți o arhivă **zip** care conține fișierul `stats` și, eventual, un `README` în care explicați ce și cum ați făcut. Dacă script-ul conține detalii suficiente, nu este nevoie să includeți un `README`.

Atenție!

- **Nu** trimiteți o mașină virtuală. Script-ul vostru poate rula la fel de bine pe orice mașină cu Bash dacă l-ați scris corect.
- **Nu** trimiteți în arhivă și checker-ul. Este mare, upload-ul va dura mult, și va fi ignorat! Testarea automată va utiliza aceeași arhivă disponibilă și vouă, cea de la `http://elf.cs.pub.ro/uso/res/uso-tema3-checker.tar.bz2`
- Pentru că arhiva pe care o trimiteți este mică, nu mai este nevoie să trimiteți întâi MD5-ul, ci doar arhiva. Dar **arhiva trebuie trimisă înainte de deadline**, nu se mai aplică sistemul cu upload-uri după deadline.

Puteți uploada mai multe versiuni ale temei pe vmchecker (în caz că nu trece toate testele din prima). Vom lua în considerare punctajul obținut de ultima arhivă. **Atenție să nu uploadați arhiva din nou după deadline, veți fi depunctați automat.**

Punctare

Nota pe întreaga temă este dată de punctajul acumulat pe toate task-urile, punctaj afișat de către programul `check` la execuția fără nici un parametru.

Se pot lua maxim **100** de puncte pe întreaga temă. Acest punctaj este echivalent cu **0.5** puncte din nota finală.

Nu este obligatorie rezolvarea tuturor task-urilor. Punctajele sunt după cum urmează:

- teste `top_contributors`: 32 de puncte;
- teste `added_lines`: 16 puncte;
- teste `deleted_lines`: 16 puncte;
- teste `touched_files`: 16 puncte;

- teste `diff_branches`: 20 puncte;

Oricare ar fi punctajul temei, la corectarea manuală puteți pierde maxim 10 puncte pentru cod duplicat sau ilizibil.

Puteți întârzia tema maxim 24 de ore, până maxim joi, 12 ianuarie, ora 23:55. Dacă întârziați, veți fi depunctați cu 20 de puncte din cele 100 aferente temei.

Nu este permis să copiați sau să partajați secțiuni de cod. Temele vor fi verificate automat împotriva copierii și plagiatul va fi pedepsit.

Tips and Tricks

- Script-ul vostru va fi rulat ca `./stats`, prin urmare va trebui să aibă o linie `shebang`. Folosiți `/bin/bash`, **nu** `/bin/sh`. Pe unele sisteme (inclusiv Ubuntu), `/bin/sh` este, de fapt, un shell mai rapid, dar cu mai puține facilități numit `dash`. You want `bash`, not `dash`.
- Pentru a depana script-ul vostru, puteți folosi `set -x`. Această comandă builtin face shell-ul să tipărească toate comenzile pe care le execută. Nu lăsați comanda în script-ul final, pentru că o să genereze mult output în plus și o să picați toate testele. Pentru mai multe opțiuni utile ale shell-ului, folosiți `help set`.
- Majoritatea task-urilor se pot rezolva folosind `git log` cu varii parametri și utilitare standard. Folosiți cu încredere `man git-log` și nu neglijați restul documentației Git. Va fi mai ușor să înțelegeți cerințele dacă înțelegeți terminologia și cazurile de utilizare ale Git.
- Dacă aveți nevoie să folosiți fișiere temporare, puteți folosi comanda `mktemp`, care creează un fișier temporar în `/tmp`; comanda garantează că nu va suprascrie un fișier deja existent, dar este reponsabilitatea voastră să ștergeți fișierul creat când nu mai aveți nevoie de el.

Lista schimbărilor