

Terminaluppgift

Pontus Haglund

October 14, 2022

1 Introduktion

Denna övning är tänkt att testa många olika verktyg som är bra att kunna när man ska arbeta i Unix-system. Det kommer vara en ganska omfattande uppgift men deluppgifterna bör kunna lösas utan större problem efter att du tagit till dig alla terminal-mail och gjort alla andra examinerande delar i TDP001.

2 Uppgift

Varje deluppgift skall lösas och dokumenteras noggrant. Du skall föra in hela dokumentationen i din blogg. Lösningen skall formateras på ett sätt som är lätt att läsa. Vanligvis är det klokt att ha alla kommandon som används på en egen rad och i fetstil. Det är också klokt att inte låta stycken vara för långa eller för breda. Vid genomläsning av den publicerade lösningen ska det tydligt framgå att ni har ingående förståelse för alla kommandon ni kört för att lösa uppgiften. Ifall kommandon upprepas flera gånger behöver kommandot endast förklaras en gång. Till exempel kommer ni behöva använda ‘cd’ och ‘ls’ mycket ofta, men det räcker att förklara vad det gör en gång. Alla delar skall lösas med terminalen, och alla steg ni tar skall finnas med. Det är inte ok att använda exempelvis caja (utforskaren i linux mint) för att navigera till olika kataloger.

2.1 Kataloger och filer

Vi ska här skapa lite kataloger och filer till en programmeringskurs.

- Skapa katalogen ~/TDP004/lab1
- Skapa en README.md i katalogen ~/TDP004 skriv någonting i filen
- Skapa ett repository på gitlab.liu.se (du kan skapat detta genom hemsidan, du behöver inte lösa just denna del i terminalen).
- Med hjälp av git init (inte clone) koppla repot till katalogen TDP004
- Knuffa upp allt innehåll i TDP004 till git

2.2 Jobba via ssh

- Anslut till ditt konto på student-domänen genom terminalen. Genomför nedanstående steg med hjälp av denna typ av anslutning i terminalen.
- Sätt upp en katalog som är kopplad till git-repot du nyss skapade.
- Dra ner alla filer etc
- Skapa en ny katalog ../TDP004/lab2/documentation utan att först skapa lab2 (hitta ett sätt att skapa båda katalogerna samtidigt).
- Skapa filen lab1.cc i katalogen lab1
- Skapa filen lab2.cc och lab2.h i katalogen lab1
- Flytta båda lab 2 filerna till katalogen lab2
- Ändra innehållet i README.md med valfri editor
- Skapa filen junk.temp i katalogen TDP004
- Skapa en git-ignore fil som ignorerar alla filer som slutar med ‘~’ och ‘temp’
- Knuffa upp alla ändringar (utom .temp-filen då) till git

2.3 Lösa en konflikt

- Du tar dig nu tillbaka till ditt eget system och löser nedanstående där.
- Du ska framkalla en merge-konflikt i git nu. Det gör vi genom att skriva ändringar i README.md innan vi hämtar hem det vi knuffade upp från student-domänen.

- När dessa ändringar är gjorda hämtar vi hem ändringarna från gitlab och löser konflikten som uppstår.

2.4 Rättigheter och skriva till fil

- Skapa filen test.py i katalogen TDP004. Skriv ett enkelt program (exempelvis hello world) och gör så filen kan köras direkt med kommandot `./test.py`
- Lägg med kommandot `'echo'` till följande kommentar i test.py: "Detta är en kommentar som skapats med echo"
- Kopiera underkatalogen lab2 så du får en ny katalog lab3
- Byt namn på alla filerna i lab3 så de har lab3 istället för lab2 i filnamnen
- Kopiera sedan katalogen lab1 från ditt konto på studentdomänen så du får en katalog som nu heter lab4. Modifiera filen i katalogen på samma sätt som i föregående steg.

2.5 Dokumentera katalogen

- Skriv katalogstrukturen, gärna som ett träd, för katalogen TDP004 till README.md (skriv inte strukturen för hand för hand, hitta ett lämpligt kommando)

2.6 Smått och gott

- Hitta alla filer som är körbara i din hemkatalog
- Filtrera dessa filer ytterligare med begränsningen att de är python-filer och går att köra direkt med `'./filnamn.py'`
- I din hemkatalog hitta och skriv ut alla rader i alla filer (inte rekursivt ned i kataloger) som skulle räknas som kommentarer i python (börjar med `#`). Detta skall inkludera dolda filer.

2.7 Lite Bashprogrammering

Du ska nu skriva ett bashscript (i detta sammanhang ett datorprogram i en .sh-fil). Uppgiften du ska lösa är att skriva ett program där användaren vid körning av programmet förväntas mata in 2 tal. Dessa tal utgör ett

intervall där första talet är starten på intervallet och andra talet är slutet på intervallet. Programmet skall sedan skriva ut hälften av de primtal som finns i det intervallet. Varannat primtal skall skrivas ut. Exempelvis i intervallet 3 till och med 19 skall talen: 3, 7, 13 och 19 skrivas ut. Denna uppgift skall lösas med bash-programmering (det är alltså inte ok att i scriptet köra exempelvis ett python-program som löser delar av problemet).