Laboration 4 — mfind

2017-10-06

Innehåll

1 Introduktion

I denna laboration ska ni skriva ett program, `mfind`, som erbjuder samma funktionalitet som UNIX-kommandot `find` om än något begränsad.

Specifikt så har vi att `mfind` ska söka efter filer, länkar och kataloger (mappar). `mfind` skall anropas på följande sätt,

```
mfind [-t type] [-p nrthr] start1 [start2 ...] name
```

Programmet funkar så att den söker reda på filer (av en *eventuellt* angiven typ), med start i mapparna `start1 ... startN`. Sökningen fortsätter i alla nåbara undermappar.

För denna laboration behöver ni **inte** skriva en fullständig laborationsrapport, däremot ska ni skriva en text där ni resonerar kring trådsäkerheten hos ert program.

Ni bör även inkludera en kort analys över huruvida fler trådar påverkar ert programs prestanda. Ni kan använda er av `time` för att ta tid på ert program,

```
lab5 git:(master): time ./mfind /pkg comsol
[raderad output]
./mfind /pkg comsol  0,53s user 3,20s system 99% cpu 3,738 total
```

Upprepa testet 10 gånger för,

- `-p = 2`
- `-p = 3`
- `-p = 4`
- `-p = 5`
- `-p = 6`
- `-p = 7`
- `-p = 8`
- `-p = 9`
- `-p = 10`

och presentera resultatet i en graf med beskrivande text[1], där ni analyserar programmets prestanda i förhållande till antalet trådar.

Gör dessa test på ett större katalogträd, testa då **inte** på home-katalogerna utan kör istället på `/pkg/` på `scratchy`.

För att ta tid på ett program utan att skriva ut programmets utdata kan ni skriva,

```
{ time [program med ev. flaggor och argument] > /dev/null 2>&1 ; }
```

Ni kan lämpligen skriva ett bash-skript för att upprepa testet flera gånger och göra en plot med (förslagsvis) GNU Octave, Matlab eller Excel då exempelvis Octave och Matlab har stöd för att läsa in data från fil.[2]

2 Syfte

Syftet med laborationen är att visa att ni kan skriva ett trådsäkert flertrådat program. Ni kommer använda er av en *iterativ* algoritm för att navigera en trädstruktur (ett filträd).[3]

[1] Testet måste upprepas flera gånger för att kompensera för variansen i exekveringstid. Ta medelvärdet på de tidsresultat ni får för ett visst antal trådar och använd det i er plot

[2] Tips: skapa en tom fil, och gör en redirect av ovanstående kommando till den filen, `{ time [mfind etc.] > /dev/null 2>\&1 ; } 2>> filnamn`.

[3] En rekursiv algoritm lämpar sig inte i detta sammanhang, då det blir svårt att fördela arbetsuppgiften på flera trådar

3 Formalia

Notera att laborationen *måste* lösas *enskilt*. Läs riktlinjer för labgenomförande för att ha en tydlig bild om vad det innebär att jobba enskilt och notera konsekvenserna som följer av att strida mot dessa regler.

Se avsnittet "Krav på hur inlämningen ska ske" för mer information om inlämningen.

3.1 Handledning

Skicka era handledningsfrågor till `5dv088-handl@cs.umu.se`. Läs denna text och även denna som referens när ni skriver era frågor. Oftast kommer ni finna att ni löser problemen själva när ni gör detta.

4 Uppgift

Uppgiften beskrivs översiktligt i avsnittet Introduktion. I detta avsnitt så redogörs formatet på den förväntade utdatan, förväntat utseende på felmeddelanden såväl som ytterligare krav på er inlämning.

4.1 Syntax

`[]` kännetecknar ett icke-obligatoriskt argument (i förhållande till slutanvändaren) till programmet. Det vill säga att ni måste oavsett implementera den funktionaliteten som beskrivs nedan.

4.1.1 `type`

`type` är en flagga som indikerar ifall programmet ska betrakta filtyp såväl som filnamn i sin sökning, `type` kan vara antingen `d`, `f` eller `l` där

- `d` står för "directory",
- `f` står för vanlig fil,
- och `l` står för "länk".

Om `type` inte anges så är standardbeteendet definierat till att söka på filnamn och ignorera filtyp.

4.1.2 `nrthr`

`nrthr` är ett heltal som anger hur många trådar som ska användas. Om argumentet inte anges så är standardvärdet 1. Ni ska låta programmets huvudtråd köra samma kod som trådarna, dvs om `nrthr = 3` så ska det skapas 2 nya trådar.

Vidare har vi att en användare ska till programmet kunna ange flera startkataloger från vilka sökningen ska göras, varvid *åtminstone en* startkatalog måste anges: `start1`.

4.1.3 `name`

`name` är namnet på den fil/mapp/länk som `mfind` ska söka efter. Er lösning behöver inte hantera wildcards (`*`, `?`) eller regex.

4.2 Utdata & Felhantering

För varje sökträff ska fulla sökvägen till den matchade "filen" (mapp/länk) skrivas ut på `stdout`. Alla påkomna fel ska skrivas ut på `stderr`.

I enlighet med GNU `find` så ska retur-statusen för `mfind` vara 0 om alla filer bearbetas utan fel, och om ett eller flera fel inträffar så ska retur-statusen vara större än noll.

Det normala beteendet för UNIX program som av någon orsak inte kan läsa en eller flera filer är att skriva ut ett felmeddelande och fortsätta med resten av filerna/katalogerna. Ni skall därför inte avsluta programmet vid första bästa problem utan hantera det snyggt, hoppa över problemområdet och fortsätta med resten av katalogerna/filerna.

4.3 Viktigt!

Låt varje tråd räkna hur många mappar den gått igenom (dvs antalet anrop till `opendir`) och skriv ut dessa antal i slutet av programmet då alla trådar terminerat. Då kan vi se om alla mappar hanterats och dessutom få en uppfattning om alla trådar fått ungefär lika mycket arbete.

Utskrifterna ska vara på formen,

```
Thread: <pthread_self()> Reads: <antalet anrop till opendir()>
```

vilket kan till exempel se ut som, (`-p` = 3),

```
Thread: 139722862274304 Reads: 20903
Thread: 139722820695808 Reads: 22052
Thread: 139722854254336 Reads: 20814
```

4.4 Algoritm

1. Lägg alla start-mappar som angetts som argument till programmet i en datastruktur, förslagvis en kö eller en stack. Lämpligen kan ni använda den länkade listan från laboration 1. Denna datastruktur innehåller de mappar som vi har kvar att traversera. När ni stöter på en ny undermapp som ni inte tidigare traverserat så lägger ni till den i datastrukturen.

2. Skapa `(nrthr - 1)` trådar, som alla exekverar samma funktion. Låt programmets huvudtråd exekvera samma funktion.
3. Trådalgoritm (grov skiss): Varje tråd ska exekvera denna enskilt tills dess att det inte finns några fler mappar att traversera

- Plocka en mapp från er datastruktur
- Läs filerna i mappen, om filen är en mapp lägg till den i datastrukturen.
- Om namnet, och ifall filtypen är angedd, matchar skriv ut sökvägen. Ni kan även lägga till alla sådana sökvägar i en lista och skriva ut dessa på slutet. Detta har för- och nackdelar.
- Inkrementera någon sorts räknare för varje mapp som tråden stötte på. Se [viktigt].
- Återgå till att plocka en mapp från datastrukturen

Reflektera över vad som krävs för att en tråd ska kunna avsluta sin exekvering. Det räcker inte med att er datastruktur är tom. Det går inte på ett trådsäkert sätt bedömma om en datastruktur är tom, varför? Ni behöver förslagsvis ett mutex eller en semafor samt en condition variable.

4.5 Ytterligare krav

- Det ska finnas med en fungerande `Makefile` som skapar en exekverbar fil som heter `mfind`.
- Använd funktionen `getopt` för att hantera flaggor från användaren, dvs. `-t` och `-p`. Vi rekommenderar att ni läser `man`-sidan för denna funktion med terminalkommandot `man -s3 getopt`. Lämpligen används den fördefinierade globala variabeln `optind` för att ta reda på vilket nummer den första parametern som inte är en flagga har.
- Länkar ska inte tolkas som directoryn, utom i det fall det är en länk angiven som `startdir`.

5 Exempelkörningar

Några exempel på hur körningar kan se ut:

```
scratchy:~/edu/sysprog/lab4> ./mfind /pkg/comsol comsol
/pkg/comsol
/pkg/comsol/bin/comsol
/pkg/comsol/5.2a/doc/html/comsol
/pkg/comsol/5.2a/configuration/comsol
/pkg/comsol/5.2a/bin/glnxa64/comsol
/pkg/comsol/5.2a/bin/comsol
```

```
scratchy:~/edu/sysprog/lab4> cd /pkg/
scratchy:/pkg> ~/edu/sysprog/lab4/mfind comsol comsol
comsol
comsol/bin/comsol
comsol/5.2a/doc/html/comsol
comsol/5.2a/configuration/comsol
```

```
comsol/5.2a/bin/glnxa64/comsol
comsol/5.2a/bin/comsol
```

```
scratchy:/pkg> cd ~/edu/sysprog/lab4
scratchy:~/edu/sysprog/lab4> ./mfind /pkg/comsol/ /usr/local/bin/ comsol
/pkg/comsol/
/pkg/comsol/bin/comsol
/pkg/comsol/5.2a/doc/html/comsol
/pkg/comsol/5.2a/configuration/comsol
/pkg/comsol/5.2a/bin/glnxa64/comsol
/pkg/comsol/5.2a/bin/comsol
/usr/local/bin/comsol
```

```
scratchy:~/edu/sysprog/lab4> ./mfind -t d /pkg/comsol/ /usr/local/bin/ comsol
/pkg/comsol/
/pkg/comsol/5.2a/doc/html/comsol
/pkg/comsol/5.2a/configuration/comsol
```

```
scratchy:~/edu/sysprog/lab4> ./mfind /opt/comsol/ /usr/local/bin/ comsol
./mfind: `/opt/comsol/': No such file or directory
/usr/local/bin/comsol
```

```
scratchy:~/edu/sysprog/lab4> ./mfind -t l /pkg/comsol/ /usr/local/bin/ comsol
/usr/local/bin/comsol
```

```
scratchy:~/edu/sysprog/lab4> ./mfind ~mr/src cdecl.c
./mfind: cannot read dir /Home/staff/mr/src: Permission denied
```

```
scratchy:~/edu/sysprog/lab4> ./mfind /Home/special/www/public_html/kurser/5DV088/HT09/lab/test index.html
/Home/special/www/public_html/kurser/5DV088/HT09/lab/test/index.html
./mfind: /Home/special/www/public_html/kurser/5DV088/HT09/lab/test/dir1: Permission denied
/Home/special/www/public_html/kurser/5DV088/HT09/lab/test/dir2/index.html
./mfind: /Home/special/www/public_html/kurser/5DV088/HT09/lab/test/dir3: Permission denied
/Home/special/www/public_html/kurser/5DV088/HT09/lab/test/dir4/index.html
```

```
scratchy:~/edu/sysprog/lab4> ./mfind /Home/special/www/public_html/kurser/5DV088/HT15/lab/lab4/dir1 /Home/spec
/Home/special/www/public_html/kurser/5DV088/HT15/lab/lab4/dir1/fil
/Home/special/www/public_html/kurser/5DV088/HT15/lab/lab4/link1/fil
```

6 Tips

- Ta en titt på `opendir` och `lstat`, även här kan ni använda er av deras `man`-sidor, då med anropen `man opendir` och `man lstat`.
- Lägg inte till mapparna `.` och `..` i er datastruktur. Varför?
- En annan viktig sak att tänka på är att bara lägga till mappar i datastrukturen ner i kataloger. Länkar till kataloger ses som länkar och dessa ska du inte göra `opendir` på. Dock finns ett viktigt undantag: Om det är en länk till en katalog som angivits som argument på kommandoraden, ska du göra `opendir` på argumentet och rekursera igenom den underliggande katalogstrukturen. Om en länk anges som argument ska länknamnet (och inte det namn den länkar till) synas i utskriften.

Det är absolut inte tillåtet att använda funktionen `ftw` som finns i ett av C:s bibliotek. Det är meningen att *ni* ska visa att ni på egen hand kan traversera en trädstruktur.

Ska ni testa er implementation av `mfind` på ett större katalogträd, testa då **inte** på home-katalogerna utan kör istället på `/pkg/` på `scratchy`.

Testa vad som händer om ditt program försöker öppna en katalog du inte har läsrättighet till. Du kan använda kommandot `chmod 000 katalognamn` för att ta bort din egen läsrättighet på en katalog. Råkar du göra kommandot på exempelvis din hemkatalog kontakta då support.

6.1 Relevanta avsnitt i kurslitteraturen

Relevanta avsnitt i "Advanced Programming in the UNIX Environment, Third Edition" inkluderar bland annat kapitel 11, specifikt så kan figurerna

- 11.10 ge er insikt i hur ni kan skydda en dynamiskt allokerad datastruktur med `mutex`,
- 11.15 användas som inledande referens för huruvida ni kan synkronisera arbete mellan trådar

men ni kan med säkerhet skippa avsnitten

- 11.6.4, 11.6.5 Reader-Writer locks,
- 11.6.7 Spin locks,
- och 11.6.8 Barriers

och kapitel 4.2, 4.23, men relevant läsning är inte exklusivt begränsad till dessa kapitel.

6.2 Globala variabler

En eller par globala variabler kan vara motiverade i detta program.

7 Inlämning

Via inlämningssidan på kursens webbsida. För inlämningsdatum se kursens webbsida.