Høgskolen i Østfold

EKSAMEN

Emnekode: ITF22514	Emne: Operativsystemer og nettverk
Dato: 1.12.14	Eksamenstid: kl. 09.00 til kl.13.00
Hjelpemidler: Alle trykte og skrevne	Faglærer: Jan Høiberg
Eksamensoppgaven: Oppgavesettet består av 6 sider, inkludert denne forsiden. Kontroller at oppgaven er komplett før du begynner å besvare spørsmålene. Oppgavesettet består av 4 oppgaver med i alt 25 deloppgaver. Les hver oppgave nøye før du begynner på besvarelsen. Alle Linux-kommandoer og shellprogrammer skal skrives i Bourne Again Shell (bash). Legg vekt på å skrive en lett forståelig besvarelse med ryddig kode.	
Sensurdato: 2.1.15 Karakterene er tilgjengelige for studenter på Studentweb senest to virkedager etter angitt sensurfrist. Følg instruksjoner gitt på: www.hiof.no/studentweb	

Oppgave 1: Prosesser, tråder og Java

Du har kompilert et CPU-intensivt Java-program `Beregn.java`, som gjør en svært tidkrevende matematisk beregning og bruker så mye CPU som det er mulig å få tilgang til. Programmet gjør ingen interaktiv I/O eller venting på data, og har tilstrekkelig med RAM tilgjengelig. Det er ingen andre prosesser på maskinen som krever mye ressurser mens dette programmet skal kjøres.

a) Skriv en Linux-kommando som starter og kjører dette ferdig kompilerte programmet i *bakgrunnen* (slik at shellet ikke må vente på at programmet skal bli ferdig).

b) Skriv et Linux shellprogram (med maks. to linjer med kode) som starter dette programmet *to* ganger, slik at det opprettes to uavhengige prosesser som begge kjører det samme programmet (tilnærmet) *samtidig*.

Resten av deloppgavene i oppgave 1 kan besvares selv om du ikke har svart på de to deloppgavene ovenfor.

De to uavhengige, samtidige prosessene som du startet i forrige deloppgave, kjører på en maskin som bare har én enkel CPU.

c) Vil et multitasking operativsystem som Linux gjøre at det går raskere å fullføre disse prosessene når de startes samtidig som i deloppgave b), sammenlignet med om de kjøres rett etter hverandre? Gi en kort begrunnelse for svaret.

Du skriver nå om Java-koden til `Beregn.java`, og lager et program som bruker to *tråder* (Java threads). De to trådene gjør til sammen den samme regnejobben som ble utført ved å starte to uavhengige prosesser som i deloppgave b) ovenfor.

d) Du starter det nye Java-programmet med to tråder på samme maskin som tidligere. Vil dette gå vesentlig raskere enn å kjøre to uavhengige prosesser som i deloppgave b)? Gi en kort begrunnelse for svaret.

e) Java-programmet med to tråder flyttes til en ny maskin med to CPU'er, som hver er like raske som den enkle CPU'en i deloppgavene ovenfor. Denne maskinen har installert Linux med støtte for multiprosessor maskinvare. Hva kan vi forvente mht. kjøretiden når programmet rekompileres og kjøres på den nye maskinen?

(Slutt på på oppgave 1)

Oppgave 2: Minnehåndtering

a) Gi en *kortfattet* beskrivelse av oppbygging og virkemåte for virtuelt minne med paging, *uten* segmentering (hver prosess får tildelt ett sammenhengende virtuelt minneområde). Tegn gjerne et par figurer for å illustrere beskrivelsen.

b) Hvorfor gir *segmentert* virtuelt minne bedre sikkerhet og reduserer muligheten for feil, sammenlignet med minne uten segmentering?

c) En *cache miss* oppstår når CPU'en ikke finner dataene den skal jobbe med i cache-minnet, og i stedet må hente inn dataene fra RAM. En *page fault* oppstår når CPU'en ikke finner dataene den trenger i RAM, og en hel page med data må hentes fra disk. Hva tror du er mest tidkrevende av en cache miss og en page fault? Er forskjellen i tid det tar stor? Begrunn svarene dine.

d) I et Java-program brukes det en integer array `A` med plass til 200 millioner heltall. Når man bruker dette arrayet vil elementene `A[0]`, `A[1]`, `A[2]` og så videre legges rett etter hverandre i RAM. Den viktigste og mest tidkrevende delen av programmet ser slik ut:

```
int i, j;
for(i = 0; i < 2000000; i++)
{
  j = i * 100;
  A[i] = i;
}
```

Du kompilerer og kjører dette programmet, og det fullføres svært raskt.

Du gjør deretter følgende endring i programmet: Indeksen endres fra `i` til `j` i linjen der verdien skrives til arrayen `A`, slik at denne linjen nå blir:

```
A[j] = i;
```

Deretter kompilerer og kjører du programmet på nytt. Det går nå mye langsommere, og bruker nesten 50 ganger så lang kjøretid.

Diskuter kort mulige årsaker til at den nye versjonen av programmet bruker mye lengre tid, til tross for at det utføres det samme antall, to millioner, skriveoperasjoner til RAM.

Hint: Bruk gjerne argumenter fra deloppgave c) ovenfor.

(Slutt på på oppgave 2)

Oppgave 3: Linux shell og shellprogrammering

a) Hvilke av disse fire kommandoene lager en kopi av filen `mintekst.txt`?

```
ls mintekst.txt > mintekst2.txt
cat mintekst.txt > mintekst2.txt
mv mintekst.txt > mintekst2.txt
cat mintekst.txt > mintekst.txt
```

b) Skriv en Linux-kommando som legger resultatet av kommandoen `pwd` i filen `/tmp/pwd`

c) Skriv en kommando som kopierer alle filer, mapper og undermapper fra katalogen `/etc` til katalogen `/mnt/backup`.

d) Hva gjør følgende kommando:

```
ps aux | grep "^ola" | wc -l
```

Hint: Se deloppgave i) nedenfor for en beskrivelse av output fra `ps aux`.

e) Systemfilen `/var/log/auth.log` er en loggfil som oppdateres automatisk av Linux. Filen beskrives slik:

```
/var/log/auth.log -- Contains system authorization information,
including user logins, login failures and authentication
mechanisms that were used.
```

Hva gjør følgende kommando:

```
cat /var/log/auth.log | grep "failure"
```

f) Manualsiden for Linux-kommandoen `tail` inneholder bl.a. følgende tekstlinjer :

```
NAME
       tail - output the last part of files
SYNOPSIS
       tail [OPTION]... [FILE]...
DESCRIPTION
       Print the last 10 lines of each FILE to standard output.

      -f  output appended data as the file grows
```

Forklar med egne ord hva kommandoen `tail -f filnavn` gjør. Gi et eksempel på en situasjon der det kan være nyttig å bruke en slik kommando

Hint: Se forrige deloppgave.

g) Skriv et shellprogram med navn arg, som tar to argumenter og skriver dem ut med ordet *og* i mellom. Eksempel på hvordan en kjøring kan se ut:

```
$ ./arg En To
En og To
```

h) Hva utføres av følgende shellprogram:

```
#!/bin/sh
ext=$1
for filnavn in *$ext
do
  mv $filnavn /tmp
done
```

i) Kommandoen `ps aux` på et Linux-system skriver ut statusinformasjon for alle prosesser på systemet. Her er noen linjer (inkludert header-linjen) med output fra denne kommandoen, kjørt på `ask.hiof.no`:

```
USER       PID %CPU %MEM  VSZ  RSS TTY  STAT START TIME COMMAND
oyvindsy  3244  0.0  0.0 16376 2024 ?     S  14:30 0:00 sshd: oyvindsy@notty
oyvindsy  3247  0.0  0.0  2324  684 ?     Ss 14:30 0:00 /usr/lib/openssh/sftp-server
mminahp   3580  0.0  0.0 16376 2024 ?     S  14:31 0:00 sshd: minahp@notty
oyvindsy  6506  0.0  0.0  2324  672 ?     Ss 14:38 0:00 /usr/lib/openssh/sftp-server
eddies    7213  0.0  0.0 16376 2008 ?     S  13:17 0:00 sshd: eddies@pts/7
eddies    7215  0.0  0.0  8852 4084 pts/7 Ss 13:17 0:00 -bash
martinso  7369  0.0  0.0 16376 2040 ?     S  13:17 0:00 sshd: martinso@notty
christsf 10431  0.0  0.0 16376 2168 ?     S  11:27 0:00 sshd: christsf@notty
christsf 10433  0.0  0.0  2324  676 ?     Ss 11:27 0:00 /usr/lib/openssh/sftp-server
janh     10858  0.0  0.0 16376 1996 ?     S  14:49 0:00 sshd: janh@pts/4
janh     10860  0.0  0.0  7940 3032 pts/4 Ss 14:49 0:00 -bash
minahp    3582  0.0  0.0  2324  660 ?     Ss 14:31 0:00 /usr/lib/openssh/sftp-server
```

Vi ser at alle linjene med prosessinformasjon begynner med navnet på brukeren som eier prosessen. Kolonnen `VSZ` angir antall kilobytes med virtuelt minne som totalt er allokert til prosessen. Kolonnen `RSS` viser hvor mye fysisk minne i RAM som prosessen bruker.

Skriv et shellprogram som tar et brukernavn som parameter, og som beregner og skriver ut de totale summene av hhv. kolonnene `VSZ` og `RSS`, for alle prosesser eid av denne brukeren. Det er ikke nødvendig å sjekke om antall parametre til programmet er korrekt, eller om parameteren har en riktig verdi.

Hint: For å behandle output fra `ps aux` linje for linje, kan det brukes en temporærfil som deretter leses inn igjen i en løkke. Oppgaven kan også løses uten bruk av en ekstra fil.

(Slutt på på oppgave 3)

Oppgave 4: Linux, servere og nettverk

a) Du ser følgende linje i apachekonfigurasjonen din. Hva gjør den?

```
Redirect permanent /ansatte http://www.nyesider.no/ansatte
```

b) Hva er forskjellen på `Redirect` og `Alias` i apache?

c) Hva er en `.htaccess` fil i apache?

d) Ta utgangspunkt i et nett tilsvarende det som har vært labben. Du har fått to nye linux filservere på det interne nettet ditt (i tillegg til den som dere satte opp). Det er bestemt at alle de tre serverene skal ha FTP-tilgang, og disse skal kunne nås utenfra det lokale nettet. Hvordan vil du sette opp `iptables` slik at du kan få tilgang til FTP på alle tre serverne fra internett?

e) Du får beskjed om at du skal sette opp en mailliste som heter:

`webansvarlige@gruppe10.no`.

Det skal være 5 lokale brukere på serveren som skal få mailen, i tillegg til 7 eksterne brukere (du har e-postadressene). Hvordan kan du få til dette i `Sendmail`?

f) Forklar hvordan Spamassassin fungerer i forhold til Sendmail på et Linuxsystem.

g) Du planlegger en ny server, og vil at alle brukerne skal ha følgende mappestruktur på sitt hjemmeområde:

```
/home/brukernavn/bilder
/home/brukernavn/dokumenter
/home/brukernavn/html
/home/brukernavn/html/index.html
```

Hvordan får du til dette når du skal legge til 4500 brukere?

(Slutt på på oppgave 4)