NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

Løsning på TDT4155 Datamaskiner og operativsystemer

Mandag 15. desember 2003, kl. 0900 - 1300

Det ønskes korte og konsise svar på hver av oppgavene.
Begrunn svar på alle oppgaver som ikke er flervalgsspørsmål.
Les oppgaveteksten meget nøye, og vurder hva det spørres etter i hver enkelt oppgave.

Dersom du mener at opplysninger mangler i oppgaveformuleringene, beskriv de antagelsene du gjør.

Datamaskiner dekkes av oppgavene 1-4 mens Operativsystemer dekkes av oppgavene 5-8.

Oppgave 5 – Operativsystemer generelt (Operating systems in general) – 12.5 %

a) Beskriv kort hva et operativsystem er

SVAR:

- Et program som kontrollerer en maskins ressurser, og som tilbyr tjenester til vanlige brukere eller andre program.

b) Diskuter kort hvilke oppgaver de ulike komponentene (functions) i et operativsystem løser

SVAR:

- CPU-bruk:
  Implementerer prosess-begrepet, synkroniserer prosesser, tildeler prosessortid til prosesser

- I/O-bruk:
  Implementerer fil-begrepet, tildeler diskplass og disktid til prosesser

- Lager-bruk:
  Implementerer virtuelt-lager-begrepet, tildeler primærlagerplass til prosesser

c) Gi eksempler på hvordan en maskins avbruddssystem (interrupt system) får betydning for dets operativsystem

SVAR:

- På en enprosessor maskin kan operativsystemet implementere gjensidig utelukkende regioner ved å slå av og på avbruddssystemet. Dette må operativsystemet implementere annerledes i flerprosessor maskiner.

d) Angi kort hvordan moderne operativsystemer skiller seg fra tidligere generasjoners systemer

SVAR:

- Moderne operativsystemer er ofte mikrokjernebaserte, objektorienterte og trådbaserte. De kan også være tilpasset flere prosessorer i tett koplete konfigurasjoner (multiprosessorer) eller løst koplete konfigurasjoner (distribuerte system).

Oppgave 6 – Bruk av CPU og I/O (CPU and I/O management) – 12.5 %

a) Beskriv kort hva som skjer i forbindelse med et prosess-skifte (process switch)

SVAR:

- CPU-en tas fra den kjørende prosess og gis til en ny prosess (muligens den samme). Begge prosessene vil endre tilstand (fra kjørende til kjørbar – og fra kjørbar til kjørende). Det trengs modusskifter inn og ut av operativsystemet (fra brukermodus til systemmodus – og fra systemmodus til brukermodus).

b) Diskuter kort fordeler og ulemper med flerprosessorsystemer (multi processor systems) kontra singleprosessorsystemer (single processor systems)

SVAR:

- Fordeler med flerprosessorsystemer:
  Økt pålitelighet, økt ytelse, reell parallellitet
  (og ikke kun simulert)

- Ulemper med flerprosessorsystemer:
  Kompleks programvare, større ytelsesgap
  (mellom prosessering og lagring/innlesing-utskrift)

c) Gi eksempler på noen operativsystemoppgaver som har innvirkning på ytelsen til filsystemer (file systems)

SVAR:

- Plassallokering på disker - og tidsstyring av disker.
  Caching og bufring av data mellom primærlager og sekundærlager.

d) Ang kort hvorfor og hvordan tidsstyring av disker (disk scheduling) skiller seg fra tidsstyring av prosessorer (processor scheduling)

SVAR:

- En disk har flere mindre enheter som kan allokeres til mange, ulike applikasjoner samtidig. (En singelprosessor kan kun allokeres til en, bestemt applikasjon på et gitt tidspunkt).

  En disk er et ikke-sekvensielt medium:
  Det betyr noe hvilken enhet på disken som aksesseres.
  (En multiprosessor er mer som et sekvensielt medium:
  Det betyr ikke noe hvilken av prosessorene som brukes).

- Eksempler på algoritmer for tidsstyring av disker:
  Først-inn-først-ut (FIFO), Korteste søk først (SSTF),
  Toveis heis (SCAN), Enveis heis (C-SCAN)

  Eksempler på algoritmer for tidsstyring av prosessorer:
  Først-inn-først-ut (FCFS), Kontinuerlig rundgang (RR),

Korteste totaltid først (SPN), Korteste gjenværende tid først (SRT),
Høyeste responsforhold først (HRRN), Tilbakekopling (FB)

- Kun FIFO / FCFS er lik.
  (Og det er ikke noen god algoritme for verken disker eller prosessorer).

Oppgave 7 – Synkronisering av prosesser (Process synchronization) – 12.5 %

a) Beskriv kort hva som kreves i forbindelse med gjensidig utelukkelse (mutual exclusion)

SVAR:

- En må implementere en absolutt gjensidig utelukkelse
- En prosess som er innenfor kan forventes å ikke forbli der for lenge
- En prosess som er utenfor skal ikke kunne forhindre andre som vil inn
- En må sikre seg mot forekomster av utsulting og / eller vranglås
- En må sikre seg mot unødige forsinkelser når flere vil inn
- En kan ikke gjøre noen antagelser om antall prosessorer,
  antall prosesser eller deres relative hastighet

b) Diskuter kort minst to prosess-synkroniseringsoppgaver som ikke kan håndteres i maskinvare men som må håndteres i programvare

SVAR:

- Gjensidig utelukkelse – der hvor den kritiske regionen er så lang at aktiv venting blir uaktuelt; dvs. passiv venting blir nødvendig
- Generell tidsordning – som for eksempel ved utveksling av spørsmål og svar mellom en klient og en tjener

c) Gi eksempler på hvordan hver av de prosess-synkroniseringsoppgavene du har diskutert over løses med meldinger (messages)

SVAR:

- Gjensidig utelukkelse:

```
Var
  Msg: Array [0..N] of Message;
  R.Mbx: Mailbox

Initialize:
Send (R.Mbx, Msg[0])

EnterCritical (R):
Receive (R.Mbx, Msg[I])
```

ExitCritical (R):
Send (R.Mbx, Msg[I])

- Generell tidsordning:

```
Var
  CM, SM: Message;
  C, S: Port

KlientProsess:
Repeat
  <Generate Question>;
  Send (S, CM);
  Receive (S, CM);
  <Apply Answer>
Forever

TjenerProsess:
Repeat
  Receive (C, SM);
  <Apply Question>;
  <Generate Answer>;
  Send (C, SM)
Forever
```

d) Angi kort hva som skiller monitorer (monitors) fra semaforer (semaphores)

SVAR:

- Semaforer er generelle verktøy, som det er lett å begå feil med
- Monitorer er spesielle verktøy, som det er lett å bevise korrekthet for

Oppgave 8 – Håndtering av lager (Memory management) – 12.5 %

a) Beskriv kort hva lokalitetsprinsippet (locality principle) betyr

SVAR:

- Lokalitetsprinsippet tilsier at lagereferanser fra et gitt program statistisk seg klomper seg sammen; dvs. neste lagerreferanse er ofte nær forrige lagerreferanse

b) Diskuter kort hvilke lagerhåndteringsoppgaver som ikke kan løses i maskinvare men som må løses i programvare

SVAR:

- De lagerhåndteringsoppgavene som forekommer relativt sjeldent, men som samtidig er relativt tidkrevende.

- For virtuelt lager vil dette si (hvor data = sider / segmenter):

  Innhenting: Når skal data som trengs leses inn i primærlageret
  Tilbakeføring: Når skal data som er endret skrives tilbake til sekundærlageret
  Plassering: Hvor skal innhentete data lokaliseres når det ikke er fullt
  Utbytting: Hvilke data skal innhentete data overskrive når det er fullt
  Mengde: Hvor mye data bør en prosess ha tilgjengelig i primærlageret
  Antall: Hvor mange prosesser bør være tilgjengelige i primærlageret

c) Gi eksempler på hvordan hver av de lagerhåndteringsoppgavene du har diskutert over løses

SVAR:

- Innhenting: Ved behov – dog før lesing; I forkant – altså ved initiering
- Tilbakeføring: Ved behov – altså ved utbytting; I forkant – dog etter skriving
- Plassering: Første; Neste; Beste; Værste (Segmenter) – Likegyldig (Sider)
- Utbytting: OPT; LRU; LFU; FIFO (Uvanlig) – U-CLOCK; UM-CLOCK (Vanlig)
- Mengde: Behold arbeidssettet (Uvanlig) – Tilpass aksessfeilraten (Vanlig)
- Antall: Tilpass aksessfeilratene samlet (Unngå både frigang og tresking)

d) Angi kort hva som skiller sidedeling (paging) fra segmentering (segmentation)

SVAR:

- Sidedeling:
  Dataene deles i flere små moduler - som er av lik størrelse.
  Gir intern fragmentering, men unngår ekstern fragmentering.

- Segmentering:
  Dataene deles i flere små moduler - som er av ulik størrelse.
  Unngår intern fragmentering, men gir ekstern fragmentering.