NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

Løsning på TDT4186 Operativsystemer

Onsdag 30. november 2005, kl. 15.00-19.00

Det ønskes korte og konsise svar på hver av oppgavene.

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.

Oppgave 1: Synkronisering av prosesser / tråder (Process / Thread Synchronization) – 20 %

a) Angi kort forskjeller mellom tråder (Threads) og prosesser (Processes)

SVAR:

Prosesser:

- Tilsvarer mer isolerte aktiviteter
- Har mer kontekst knyttet til seg
- Kan deles i tråder

Tråder:

- Tilsvarer mer samhørende aktiviteter
- Har mindre kontekst knyttet til seg
- Kan samles i prosesser

b) Beskriv og diskuter noen konkrete tilfeller hvor tråder er mer anvendelig enn prosesser

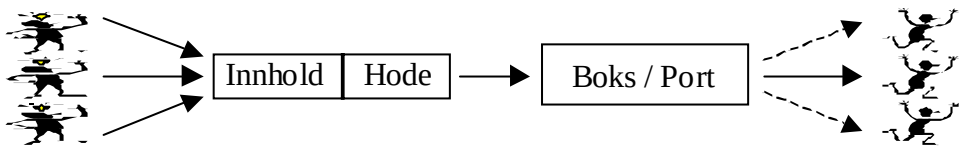
SVAR:

- Ved naturlig parallellitet innen program / applikasjon
  Eks: Med vektor/matrise orientering
- Ved flere parallelle inn-enheter
  Eks: I.f.m. tastatur og mus

c) Angi kort hva meldingsutveksling (Message Passing) er og hvordan meldingsutveksling brukes i synkroniserings sammenheng (Process / Thread Synchronization)

SVAR:

Meldingsutveksling er en kommunikasjonsform. Den kan brukes både i sentraliserte og distribuerte systemer. Basiskonseptene er et dataobjekt, melding, med to tilhørende dataoperasjoner, send og receive.



I synkroniserings sammenheng brukes en gitt melding som et token. Denne meldingen sendes fram og tilbake mellom prosesser / tråder. En gitt prosess / tråd har tillatelse til å utføre visse aktiviteter kun når den er i besittelse av token meldingen.

d) Beskriv og diskuter helt konkret hvordan meldingsutveksling kan implementeres med semaforer (Semaphores)

SVAR:

Datastrukturer – inkludert initialisering	***Dataoperasjoner***
Var Slot: Semaphore := N; Buffer: Semaphore := 1; Port: Array [1..M] of Semaphore := M*0; BufferQ: List of Item; PortQ: Array [1..M] of List of Item; **For** I := 1 **To** N <Link Item into BufferQ>	***Send (Message, Destination):*** Wait (Slot); Wait (Buffer); <Unlink Item from BufferQ>; <Copy Message into Item >; <Link Item into PortQ[Destination]>; Signal (Port[Destination]); Signal (Buffer) ***Receive (Message, Source):*** Wait (Port[This]); Wait (Buffer); <Unlink Item from PortQ[This]>; <Copy Message from Item >; <Link Item into BufferQ >; Signal (Slot); Signal (Buffer)

Oppgave 2: Bruk av lager (Memory Management) – 20 %

a) Angi kort hvilke aspekter av virtuelt lager (Virtual Memory) som kan håndteres i programvare (Software) og hvilke aspekter som må håndteres i maskinvare (Hardware)

SVAR:

Programvare:

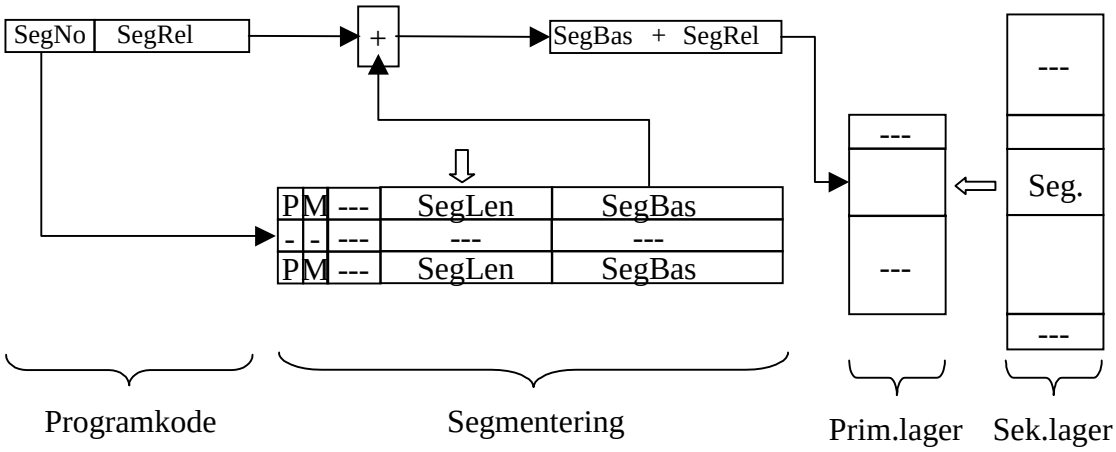
- Innhenting - når ? (ved aksess)
- Tilbakeføring - når ? (ved endring)
- Plassering - hvor ? (hvis ikke fullt)
- Utbytting - hvor ? (hvis fullt)
- Mengde - hvor mye til hver enkelt ? (responstidsfokusert)
- Antall - hvor mange totalt sett ? (gjennomstrømningsfokusert)

Maskinvare:

- Avbildning: Oversette adresser
- Sjekking: Kontrollere adresser
- Reagere med avbrudd: På manglende “info”
- Hjelpe via caching: Med viktig “info”

b) Illustrer og beskriv hvordan adresseberegning (Address Mapping) skjer ved segmentering (Segmentation)

SVAR: *Segmentering*

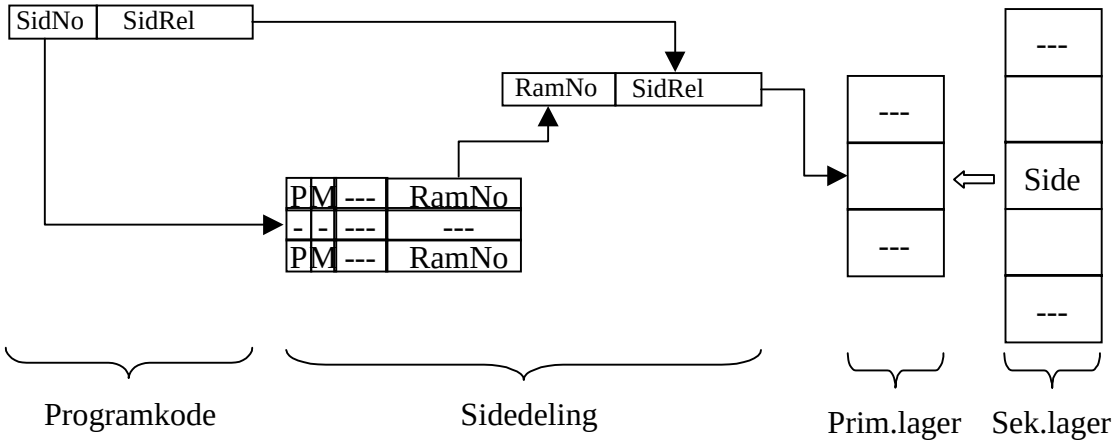


Obs1: Hvis sjekken på grenseoverholdelse (SegLen) tilsier at stedet er utenfor segmentet, blir det generert avbrudd

Obs2: Hvis sjekken på tilstedeværelse (P) tilsier at segmentet ikke er i primærlageret, blir det først overført fra sekundærlageret

c) Illustrer og beskriv hvordan adresseberegning (Address Mapping) skjer ved sidedeling (Paging)

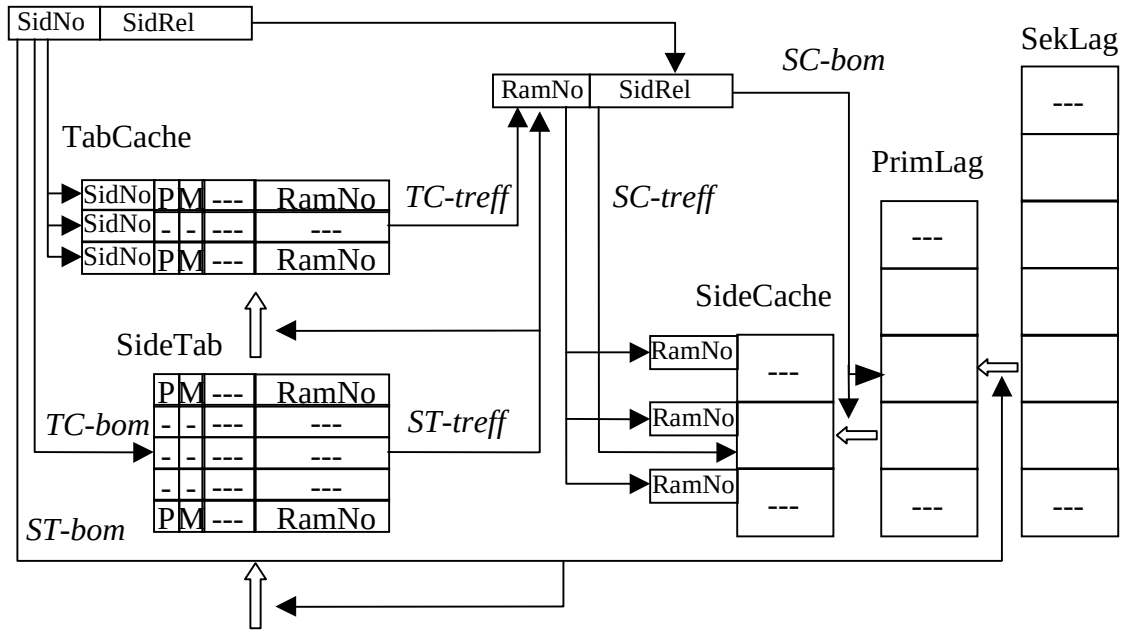
SVAR: *Sidedeleing*



Obs1: Hvis sjekken på tilstedeværelse (P) tilsier at siden ikke er i primærlageret, blir den først overført fra sekundærlageret

d) Diskuter kort bruk av nærlagring (Caching) av ulike data/info i denne sammenheng

SVAR: *Caching av side/segment tabeller samt program-kode/data (illustrert med sidedeling)*



Obs1: Ved treff i et gitt søk fortsettes til neste nivå
Obs2: Ved bom i et gitt søk fortsettes til alternativ variant
Obs3: ST-treff gir en ekstra overføring mellom SideTab og TabCache
Obs4: SC-bom gir en ekstra overføring mellom PrimLag og SideCache
Obs5: ST-bom gir en ekstra overføring inn i SideTab,
samt en ekstra overføring mellom SekLag og PrimLag

Oppgave 3: Tidsstyring av prosesser / tråder (Process / Thread Scheduling) – 20 %

a) Angi kort relevante mål (Objectives) for tidsstyring (Process / Thread Scheduling)

SVAR:

	Resultatfokus	***Systemfokus***
Ytelses-orientering	Lav responstid Lav gjennomløpstid Overholdelse av tidsfrister	Høy gjennomstrømning Høy ressursutnyttelse Lav overhead
Ikke ytelses-orientering	Unngåelse av utsulting Høy forutsigbarhet Sikring av ressursbibehold	Høy rettferdighet Oppnåelse av ressursbalansering Sikring av prosessprioritering

b) Beskriv spesifikt hvorfor andre tidsstyrings algoritmer blir nødvendige for multiprosessorer (Multi CPUs) enn for singleprosessorer (Single CPUs)

SVAR:

Med # Prosessorer $\geq$ 2:

- Det å unngå at en prosess / tråd legger beslag på en prosessor blir mindre viktig
- Det å oppnå at gitte tråder / prosesser kjører samtidig på flere prosessorer blir mer viktig

c) Beskriv likheter og ulikheter mellom noen slike multiprosessor algoritmer

SVAR:

Lastdeling: • Selvtilordning av prosessorer • Sikrer utnyttelse av prosessorer • Vanlige algoritmer via felleskø • Mulig flaskehals med felleskø • Ikke tråd på samme prosessor (ref. caching) • Ikke tråder sammen på prosessorer (ref. ytelse)	***Prosessorholding:*** • Flere/alle tråder innen en prosess får og beholder en prosessor • Sikrer samkjøring og ferdigkjøring av tråder • Ingen trådskifter ved koordinering • Maks ytelse - min utnyttelse • Singleprogrammering: CPU-tildeling tilpassbar til arbeidssett av prosessorer
Samkjøring: • Flere/alle tråder innen en prosess får en prosessor • Sikrer samkjøring av tråder • Færre trådskifter ved koordinering • Mer ytelse - mindre utnyttelse • Multiprogrammering: CPU-tildeling tilpassbar til trådantall i applikasjon	***Antallsvariasjon:*** • Antall tråder innen en prosess kan variere dynamisk • Antall prosessorer til en prosess kan variere tilsvarende • Program og system kan og bør samhandle om tilordning tråder-prosessorer • Singleprogrammering: CPU-tildeling tilpassbar til treskemulighet for prosessorer

d) Diskuter kort noen ytterligere konsekvenser av å gå fra singleprosessorer til multiprosessorer i operativsystem sammenheng

SVAR:

Synkronisering:

- Avbruddsblokkering (som for enkeltprosessorer) blir ikke mulig lenger

Lagerhåndtering:

- Hastighetsbegrensningen (sammenlignet med prosessorkapasiteten) blir økt ytterligere

Oppgave 4: Bruk av I/O (I/O Management) – 20 %

a) Angi kort hvorfor vi trenger henholdsvis mellomlagring (Buffering) og nærlagring (Caching) i forbindelse med innlesing/utskrift (I/O – Input/Output)

SVAR:

Buffering:

- Unngå blokkering i.f.m. I/O-bruk m.h.t. kjøring / swapping av prosesser
- Utjevne variasjon i I/O-bruk innen / mellom prosesser

Caching:

- Utnytte lokalitetsprinsipp – for data som for program

b) Diskuter kort ulike varianter av mellomlagring i denne sammenheng

SVAR:

Uten bufring: • Gir kjøre-blokkering • Gir swapp-blokkering	***Dobbel bufring:*** • Utjevner mindre hastighetsvariasjoner
Single bufring: • Unngår kjøre-blokkering • Unngår swapp-blokkering	***Sirkulær bufring:*** • Utjevner større hastighetsvariasjoner

c) Diskuter kort ulike algoritmer for håndtering av nærlagring i denne sammenheng

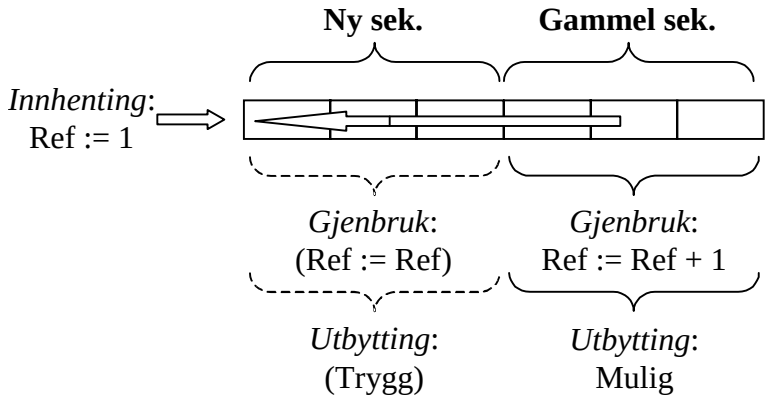
SVAR:

LRU - Minst nylig referert: • Krav: Tidsmerke til hver side til en prosess • Utbytting: Lengst ureferert	***FBS_{II} – Frekvensbasert stakk:*** • Krav: Todelt stakk for hver prosess • Utbytting: LRU+LFU på 1/2 av stakken
LFU - Minst ofte referert: • Krav: Ref.teller for hver side til en prosess • Utbytting: Sjeldnest referert	***FBS_{III} - Frekvensbasert stakk:*** • Krav: Tredelt stakk for hver prosess • Utbytting: LRU+LFU på 1/3 av stakken

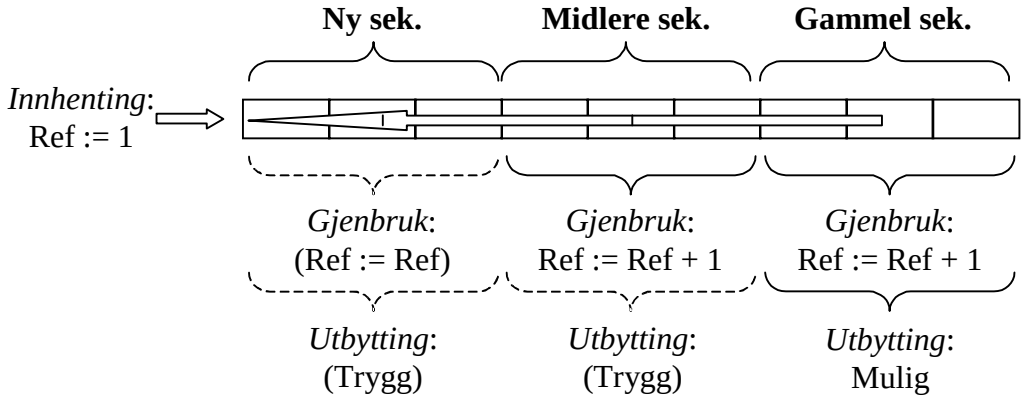
d) Beskriv algoritmen for ”Frekvensbasert stakk” (FBS – Frequency Based Stack) – og sammenlign den med algoritmene for ”Minst nylig referert” (LRU – Least Recently Used) og ”Minst ofte referert” (LFU – Least Frequently Used)

SVAR:

Todelt stakk:
LRU ordning av sider over de to deler – fra kortest til lengst tid siden ref.
Referanseteller økes kun ved treff i gammel seksjon
Utkastelse på LFU skille innen gammel seksjon kun – LRU ved likhet igjen



Tredelt stakk:
LRU ordning av sider over alle tre deler – fra kortest til lengst tid siden ref.
Referanseteller økes kun ved treff i midlere eller gammel seksjon
Utkastelse på LFU skille innen gammel seksjon kun – LRU ved likhet igjen



Sammenligning:
Todelt FBS – Tilnærmet likeverdig med LRU / LFU hver for seg
Tredelt FBS – Vesentlig bedre enn LRU / LFU hver for seg

Oppgave 5: Distribuerte systemer (Distributed Systems) – 20 %

a) Angi kort hva et distribuert system er

SVAR:

Et distribuert system er en samling av sammenkoplete, selvgående systemer. Flere fysisk adskilte systemer fremstår som ett logisk samhørende system. Delsystemene mangler felles klokke og felles lager. En felles oppfatning av tid, tilstand og verdier må således implementeres. Totalresultatet blir et partnerskap mellom delvis samvirkende, delvis uavhengige systemer.

b) Angi kort fordeler og utfordringer med distribuerte system

SVAR:

Fordeler:

- Deling av ressurser
  - Aksesserbarhet (En applikasjon mot flere ressurser og flere applikasjoner mot en ressurs)
- Ytelse på applikasjoner
  - Parallellitetsutnyttelse (Innen en applikasjon og mellom flere applikasjoner)
- Feiltoleranse
  - Pålitelighet (Fragmentering av applikasjoner og ressurser)
  - Tilgjengelighet (Replisering av applikasjoner og ressurser)
- Systemfokus
  - Større kapasitet
  - Bedre økonomi
  - Stegvis utvidelse
  - Ressurstilpasning i.h.t. organisasjonsforhold
  - Applikasjonstilpasning i.h.t. utførelsesforhold

Utfordringer:

- Levende systemer
  - Trenger justerbarhet over lang tid
  - Trenger utvidbarhet i stor skala
- Åpne systemer
  - Behov for portabilitet av ulike applikasjoner
  - Behov for interoperabilitet mellom ulike systemer

c) Diskuter kort noen eksisterende modeller (Models) for distribuerte system

SVAR:

Her forventes det ikke mer enn anslagsvis to av for eksempel de fire nedenforstående aspektene, som ofte kombineres med mellomvare til en modell.

Mellomvare lag

- En konstruerer et programvare lag (med et generelt API) mellom globale applikasjoner og lokale ressurser slik at en gitt applikasjon kan nå en vilkårlig lokal ressurs, og en gitt ressurs kan nås av en vilkårlig applikasjon – hvor mellomvare laget står for ruting av forespørslene til riktig node og oversetting av forespørslene til riktig språk

Klient-tjener aspektet

- Noen komponenter forespør aktiviteter, og noen komponenter utfører aktiviteter
- En vilkårlig komponent kan spille begge rollene rekursivt
- Hver komponent kan allokeres til den node som passer best

Gruppe-kommunikasjons aspektet

- Forespørsler skal ofte gå til en gruppe av komponenter
- Ulike former for atomiskhet innen en slik forespørsel kan kreves
- Ulike former for ordnethet mellom flere slike forespørsler kan kreves

Objekt-orienterings aspektet

- Aktiviteter knyttes ofte til metoder i objekter
- Fjerne objekter tilknyttes ofte lokale stand-in objekter

Strøm-orienterings aspektet

- Aktiviteter er ofte kontinuerlige, og ikke bare diskret, aktiviteter
- Aktiviteter er ofte langvarige, og ikke bare kortvarige, aktiviteter

d) Diskuter kort noen eksisterende standarder (Standards) for distribuerte system

SVAR:

Her forventes det ikke mer enn anslagsvis tre av for eksempel de syv nedenforstående standarder.

ISO/RM-ODP

* OO-type, rammeverk-nivå
+ Omfatter det meste som trengs
- Enormt begrepsmessig apparat

OMG/CORBA

* ORB-type, arkitektur-nivå
+ OO-basert
- Mye å sette seg inn i

OSF/DCE

* RPC-type, arkitektur-nivå
+ Enkel å ta i bruk
- Ikke OO-basert

MS – DCOM/OLE

* RPC/OO-mellomting, blanding av alt
+ MS støtter det
- MS bestemmer alt

X/OPEN – TX/XA

* DTP-fokus, blankt på mye
+ Tidlig fokus på transaksjoner, således litt oppdragende på andre
- Inkludert i andre etter hvert, således litt overflødig i seg selv

TINA

* Telekom-rettet, spesialisering av RM-ODP
+ Effektivt verktøy i sin valgte nisje
- RM-ODP++, for mye av det gode !

MSS

* Multimedia-rettet, spesialisering av CORBA
+ Effektivt verktøy i sin valgte nisje
- CORBA++, for mye av det gode ?