NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

Løsning på TDT4186 Operativsystemer

Mandag 17. desember 2007, kl. 09.00-13.00

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.

Hver av de seks oppgavene teller like mye, og hver av deloppgavene teller like mye.

Oppgave 1: Operativsystemer (Operating Systems)

a) Angi kort hva et operativsystem er

SVAR:

Et operativsystem er programvare som skal gjøre det enklere og mer effektivt å bruke maskinvaren – og derigjennom tilby ulike tjenester til brukerne eller andre programmer og kontrollere de ulike ressursene i maskinvaren og programvaren.

b) Diskuter kort hvordan mer moderne operativsystemer skiller seg fra noe eldre operativsystemer

SVAR:

Et mer moderne operativsystem er ofte mikrokjernebasert, blir ofte objektorientert, bruker gjerne tråder, går gjerne på flere CPU-er og kan ofte gjerne være inkludert som en komponent i et større distribuert system.

Oppgave 2: Synkronisering av prosesser (Process Synchronization)

a) Angi kort hva vi trenger synkronisering til

SVAR:

Synkronisering trengs for å håndtere parallellitet mellom ulike programmer (prosesser/tråder) eller mellom et program (prosess/tråd) og noe maskinvare (generell ressurs). Formålet er å unngå uheldige effekter av slik parallellitet.

b) Diskuter forskjellene mellom verktøyene semaforer (Semaphores), monitorer (Monitors) og meldingsutveksling (Message Passing) i forbindelse med synkronisering

SVAR:

Alle tre er generelle verktøy ved at hvert av dem kan brukes til å løse en vilkårlig synkroniseringsoppgave. Således kan hvert av dem brukes til å implementere hver av de andre to.

Men verktøyene er tilpasset ulike formål. Således er semaforer generelle – på et forholdsvis lavt implementasjonsnivå (og dermed lett å begå feil med), mens monitorer er spesielle – med

direkte fokus på gjensidig utelukkelse (og derigjennom lett å bevise korrekt bruk for), og meldingsutveksling også kan brukes for distribuerte systemer – altså uten delt lager.

c) Beskriv helt konkret hvordan meldingsutveksling kan implementeres med semaforer

SVAR:

Meldingsutveksling implementert med semaforer

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 3: Håndtering av lager (Memory Management)

a) Angi kort hvorfor vi trenger lagerhåndtering

SVAR:

Lagerhåndtering trengs for å håndtere (samtidige) programmers (prosessers/tråders) ønske om bruk av (større / mindre) primærlagerområder. Formålet er å optimalisere bruken av primærlagerområdene slik at de tilhørende programmer (prosesser / tråder) kan kjøres på en effektiv og beskyttet måte.

b) Diskuter forskjellene mellom algoritmene minst-nylig-referert (Least Recently Used), minst-ofte-referert (Least Frequently Used), 2-sjansers klokkebasert (U-Clock) og 4-sjansers klokkebasert (UM-Clock)

SVAR:

Alle fire lagerstyringsalgoritmer prøver å optimalisere et sett av flere lageraksesser ut fra et gitt kriterium. De skiller seg på hvilken eksisterende side som vil byttes ut når en ny side må hentes inn.

Minst-nylig-referert velger den siden som det er lengst tid siden har vært referert, og til det kreves et tidsmerke for hver side til en prosess. Mens minst-ofte-referert velger den siden som har vært referert færrest ganger i løpet av en periode, og til det kreves en referanseteller for hver side til en prosess.

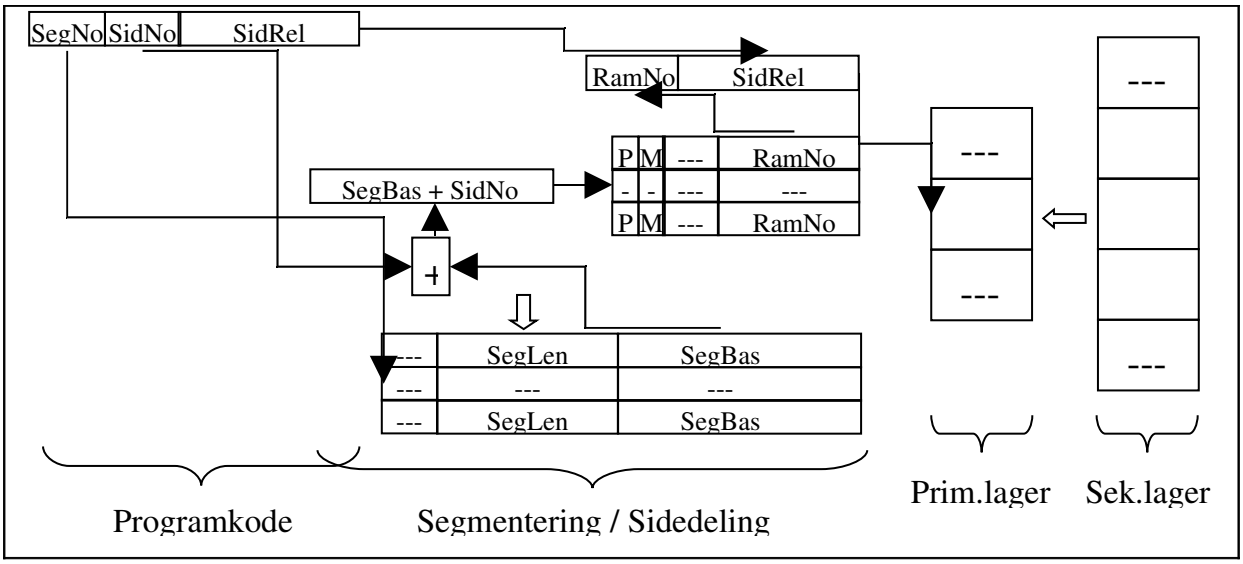
2-sjansers klokkebasert bruker et bitt for hver side til en prosess. Referert-bittet settes ved hver aksessering av siden og resettes ved forsøk på å finne en side som kan velges (ved en sekvensiell gjennomgang av alle). Således får en vilkårlig side to sjanser til å unngå å bli utvalgt. Mens 4-sjansers klokkebasert bruker to bitt for hver side til en prosess. Endret-bittet settes ved oppdatering av siden, og Referert-bittet settes ved hver aksessering av siden og resettes ved forsøk på å finne en side som kan velges (igjen ved en sekvensiell gjennomgang av alle). Derigjennom får en vilkårlig side fire sjanser til å unngå å bli utvalgt.

Minst-nylig-referert og minst-ofte-referert krever mer overhead enn 2-sjansers klokkebasert og 4-sjansers klokkebasert så vel plassmessig som tidsmessig.

c) Beskriv helt konkret hvordan segmentering (Segmentation) og sidedeling (Paging) kan kombineres i en implementasjon av virtuelt lager (Virtual Memory)

SVAR:

Virtuelt lager implementert som en kombinasjon av segmentering og sidedeling



- SegLen er angitt i antall Sider – som SidNo sjekkes mot
- Present bit P (og Modified bit M) sjekkes på Sidenivå – og ikke på Segmentnivå
- Hvis den utpekte Side ikke er i Primærlageret, må evt. en eksisterende i Primærlageret byttes ut med den nye fra Sekundærlageret gjennom en lagerstyringsalgoritme

Oppgave 4: Tidsstyring av prosesser (Process Scheduling)

a) Angi kort hvorfor vi trenger tidsstyring

SVAR:

Tidsstyring trengs for å håndtere parallelle programmers (prosessers/tråders) ønske om bruk av en / flere CPU-er. Formålet er å optimalisere bruken av CPU-ene i henhold til flere (ikke nødvendigvis samvirkende) parametre som responstid, gjennomløpstid, gjennomstrømning og ressursutnyttelse etc.

b) Diskuter forskjellene mellom algoritmene kontinuerlig-rundgang (Round Robin), korteste-totaltid-først (Shortest Process Next), korteste-gjenværende-tid-først (Shortest Remaining Time) og høyeste-responsforhold-først (Highest Response Ratio Next)

SVAR:

Alle fire tidsstyringsalgoritmer prøver å optimalisere et sett av flere CPU-behov ut fra gitte kriterier. De skiller seg på hvilket program (prosess / tråd) som velges som det neste som skal kjøres – og når slike valg kan gjøres.

Kontinuerlig-rundgang velger med faste mellomrom den prosess / tråd som har lengst ventetid siden forrige valg, mens korteste-totaltid-først velger ved hver ferdigkjøring den prosess / tråd som har det minste totale CPU-behov. Korteste-gjenværende-tid-først velger ved hver nyankomst den prosess / tråd som har det minste gjenværende CPU-behov, mens høyeste-responsforhold-først velger ved hver ferdigkjøring den prosess / tråd som har høyest brøk mellom [Ventetid + Totalt CPU-behov] og [Totalt CPU-behov].

Korteste-totaltid-først og høyeste-responsforhold-først tillater ikke avbryting, mens kontinuerlig-rundgang og korteste-gjenværende-tid-først benytter avbryting. Kontinuerlig-rundgang prioriterer ingen program, mens korteste-totaltid-først og korteste-gjenværende-tid-først prioriterer korte program, og høyeste-responsforhold-først prioriterer så vel korte som lange program.

c) Beskriv helt konkret hvordan periodebasert-tidsstyring (Rate Monotonic Scheduling) virker

SVAR:

Periodebasert-tidsstyring ordner aktiviteter i henhold til lengden på deres tilhørende perioder, og prioriteter aktivitetene heretter – med høyest prioritet til de med kortest periode, og lavest prioritet til de med lengst periode. Periodene må altså være kjent på forhånd, og det er en vurderingsbasert metode – ikke en innsatsbasert metode. Periodebasert-tidsstyring brukes gjerne for sanntidssystemer.

Oppgave 5: Håndtering av I/O (I/O Management)

a) Angi kort hva vi trenger I/O håndtering til

SVAR:

I/O håndtering trengs for å håndtere (samtidige) programmers (prosessers/tråders) ønske om bruk av (ulike) inn/ut-enheter. Formålet er å optimalisere bruken av inn/ut-enhetene slik at de tilhørende programmer (prosesser / tråder) kan kjøres på en effektiv og koordinert måte.

b) Diskuter forskjellene mellom mellomlagring (Buffering) og nærlagring (Caching) i forbindelse med I/O håndtering

SVAR:

Mellomlagring fokuserer på optimalisering av en enkelt aksess, mens nærlagring fokuserer på optimalisering av flere ulike aksesser.

Mellomlagring unngår blokkeringer og utjevner hastighetsvariasjoner, mens nærlagring utnytter lokalitetsprinsippet for så vel data som program.

c) Beskriv helt konkret hvordan enveis-heis (C-Scan) algoritmen virker

SVAR:

Enveis-heis håndterer I/O forespørsler mot en disk ved å betjene dem i samme rekkefølge som deres tilhørende spor har på disken – men kun i den ene retningen av diskens sett av spor. Når hodet har nådd diskens siste spor, hopper det således tilbake til diskens første spor igjen før forespørsler fortsettes å betjenes. Forespørsler innen ett og samme spor håndteres etter hvert som sporets blokker roterer under hodet – altså i lineær rekkefølge.

Oppgave 6: Distribuerte systemer (Distributed Systems)

a) Angi kort hva et distribuert system er

SVAR:

Et distribuert system er en samling av sammenkoplete, selvgående systemer – som omfatter flere fysisk adskilte systemer men som man ønsker skal fremstå som ett logisk samhørende system.

b) Diskuter kort hvordan distribuerte systemer skiller seg fra operativsystemer

SVAR:

Et distribuert system omfatter både maskinvare og programvare i motsetning til et operativsystem som bare omfatter programvare. Det inneholder komponenter på hver av de deltagende nodene som implementerer klassiske operativsystemelementer som prosess-sykronisering, lagerhåndtering, tidsstyring og I/O-håndtering. I tillegg implementerer et distribuert systems komponenter ulike felles tjenester på tvers av de ulike nodene – slik som globale filsystemer og databasesystemer, distribuert pålitelighet og sikkerhet, samt navnetjenester og multimediatjenester.