Norges teknisk-naturvitenskapelige universitet

Institutt for datateknikk og informasjonsvitenskap

Løsningsforslag til
EKSAMENSOPPGAVE I FAG TDT4186 – OPERATIVSYSTEMER

Faglig kontakt under eksamen: Orestis Gkorgkas

Tlf.:

Eksamensdato: **11. august 2012**

Eksamenstid: **09.00-13.00**

Tillatte hjelpemiddel: **D: Ingen trykte eller håndskrevne hjelpemiddel tillatt. Bestemt, enkel kalkulator tillatt.**

Språkform: Bokmål

Sensurdato: 1. september 2012

Oppgaven er kontrollert av professor Kjell Bratbergsengen.

Oppgave 1 – Generelt – 15 %

a)

Operativsystemet tilbyr sine tjenester enten via bibliotekskall eller ved at det har en prosess som betjener tjenesten. Brukerprosessen vil kalle en biblioteksfunskjon som igjen må gjøre et systemkall. Når systemkallet skjer blir konteksten for CPUen byttet til kjernemodus slik at operativsystemet kan beskytte den ressursen som tjenesten gjør bruk av, for eksempel for hindre at brukerprosessen ødelegger ei fil.

b)

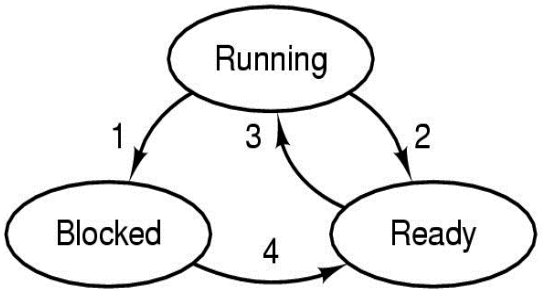
En TRAP-instruksjon brukes for å implementere et systemkall og det er en instruksjon som skifter fra brukermodus til kjernemodus. OS-kjernen vil så utføre en funksjon som er registrert for det gjeldene nummeret som er lagt inn i et register osv. Se side 49 i Tanenbaum.

c)

Fork, waitpid, execve og exit er nevnt i Tanenbaum.

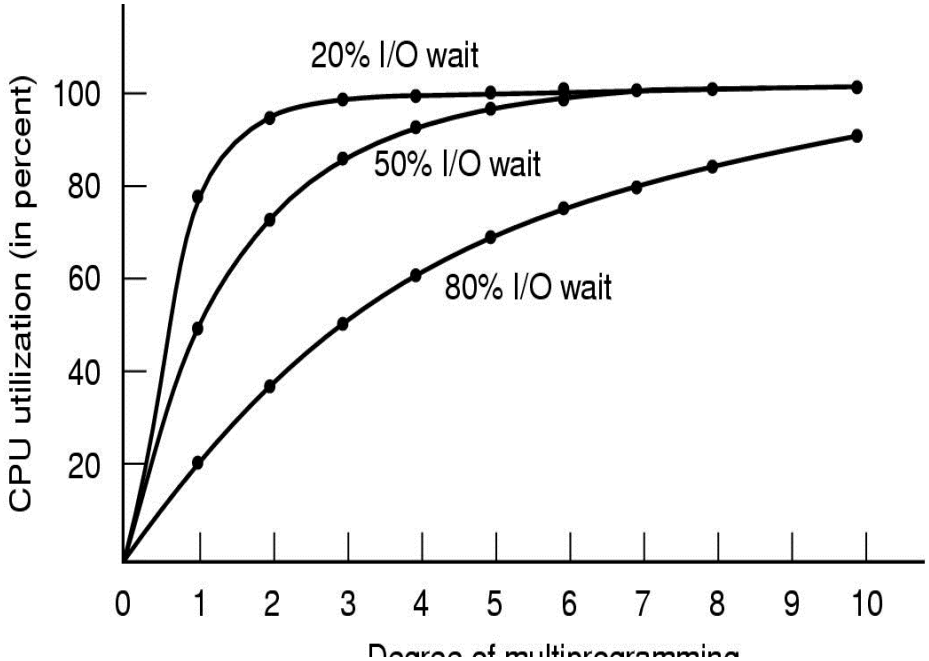
Oppgave 2 – Prosesser – 10 %

a) Prosesser har tre hovedtilstander: *Running*, *Blocked* og *Ready*. Lag en figur som viser hvordan en prosess skifter mellom de tre tilstandene.



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

b) Figuren under viser CPU-utnyttelsen som en funksjon av multiprogrammeringsgraden. Forklar kort hva figuren viser:



Figuren viser at du må ha mange prosesser/tråder som kjører for å utnytte CPU-en helt. Jo mer I/O prosessene gjør, jo flere prosesser kan CPUen betjene samtidig.

Oppgave 3 – Tråder – 10 %

Det er vanlig å anbefale å bruke (kjerne-)tråder som kan dele adresserom og ressurser. Man må da håndtere samtidig aksess av minne, men hvis man har en mottakertråd som starter de andre trådene, kan den koordinere hvilke deler av adresserommet hver tråd har lov å bruke. Enkelttrådete prosesser er tunge å bruke når man skal betjene mange klienter samtidig, fordi hver prosess har et stort overhead med ressurser den trenger. Aynkron-I/O og tilstandsmaskiner kan være vanskelig å programmere og vedlikeholde.

Oppgave 4 – Synkronisering – 20 %

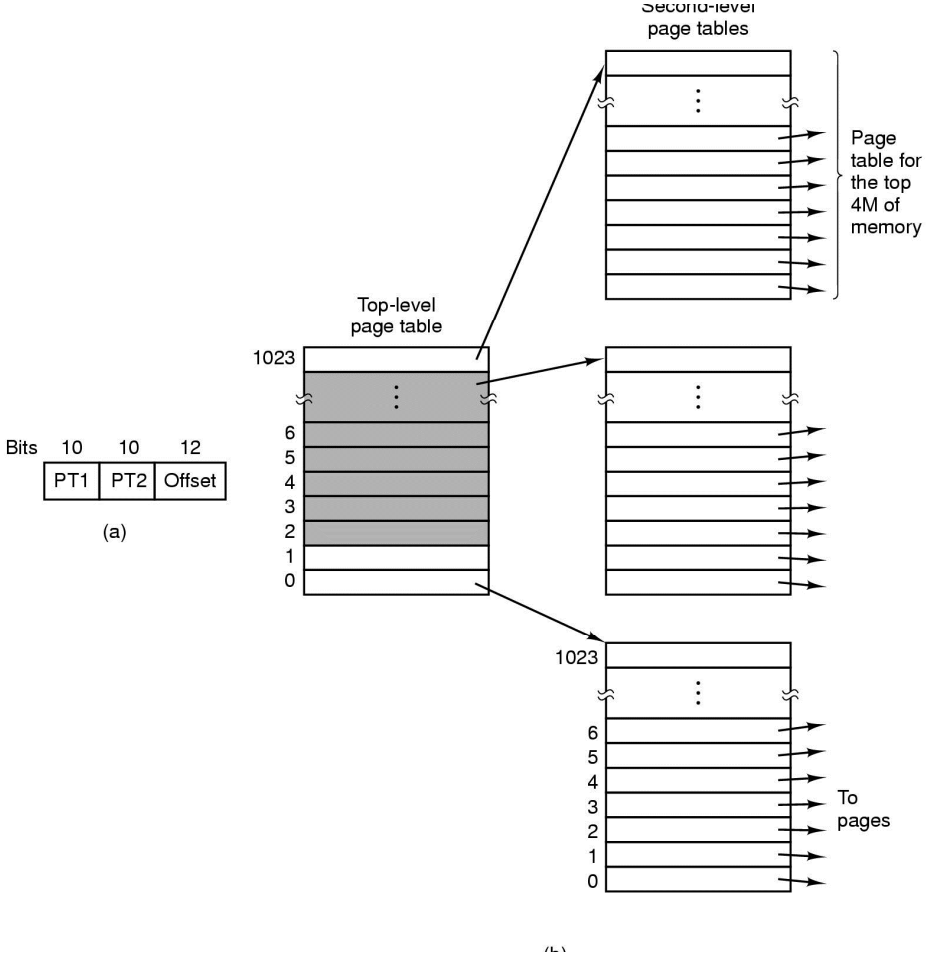
a) Her går det an å bruke forskjellige mekanismer. For eksempel mutexer, semaforer eller språkmekanismer som monitorer. Det mest naturlige er bruke den mekanismen som er tilgjengelig i den omgivelsen man programmere i. Mutex'er er enkle låser for å sørge for at en tråd om gangen har tilgang til et bestemt minneområde. Semaforer er omtrent som mutex'er, men har i tillegg muligheten til å telle hvor mange som skal ha aksess. De er dermed litt mer avanserte og kan være vanskeligere å bruke. Monitorer er laget for å være enklere å bruke, men de krever en nøyaktighet i bruken de også for å være sikker på at de ikke blir «race conditions».

b) Programmet kan få problemer fordi «guarden» er delt i to setninger som det kan komme prosesskifte mellom. For eksempel hvis consumer først kjører If (count==0),

som slår til og så får vi et prosesskifte hvor count blir inkrementert av producer som så vekker opp consumer som igjen kjører videre med sleep. Da kan consumer vente evig fordi producere har gjort sitt wakeup.

Oppgave 5 – Minnehåndtering – 20 %

a) Forklar hvordan man kan organisere en sidetabell slik at den kan støtte store adresserom. Bruk gjerne en figur.



Det brukes gjerne en flernivå sidetabell, gjerne med fire nåver. I figuren over er det brukt tre nivår, hvor de to første delene av adressen er indikser til sidetabellene i to nivåer, og den siste delen er offset innen siden. Man slår da først opp i øverste nivå av sidetabellen (ved PT1) og finner neste del av sidetabellen som man så slår opp i (ved PT2). Så bruker man offset innen den sida man finner.

b) Working set model. Se Tanenbaum side 207-211.

c) En adresse er delt opp i segment-selektor og Offset. Ved hjelp av segmentselektoren finner man segmentbeskrivelsen som inneholder bl.a. baseadressen for segmentet. Da er det å addere offset til baseadressen og man får den endelige adressen.

Oppgave 6 – Filsystemer – 10 %

a) For hver direktepeker har man en blokk, dvs. 10 * 4K.

For hver enkeltindirektepeker har vi en blokk med pekere: (4096/4)=1024 pekere til 4K blokker, dvs. 4096 K = 4 M

For hver dobbelt indirektepeker har vi $(4096/4)^2$ pekere til 4K blokker.

For hver trippelt indirektepeker har vi $(4096/4)^3$ pekere til 4K blokker.

Til sammen $(10+1024+1024^2+1024^3)$ blokker av 4K.

b) Her står det en del i Tanenbaum (side 306-307) om at det er lurt å ikke skrivecache metadata for filsystemet, dvs. alle blokker som ikke er vanlige data,

Oppgave 7 – Vranglås – 15%

a) Pre-emption

Rollback

Killing processes.

b) Her kan P2 få det den spør etter. Etter at P2 har kjørt får vi tilgj.vektor: (2 2 2 0)

Da kan P1 få det den spør etter. Etter at P1 har kjørt får vi tilgj.vektor: (4 2 2 1)

Da kan P0 kjøre og alle tre er ferdige.

Systemer er i en sikker tilstand.