NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

Løsning på TDT4186 Operativsystemer

Tirsdag 14. desember 2004, 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.

Oppgave 1: Bruk av CPU og I/O (CPU & I/O Management) – 20 %

a) Forklar i hvilke tilfeller en bør bruke henholdsvis prosesser (Processes) og tråder (Threads).

SVAR:

- Tråder kan brukes når en trenger mer samhørende delaktiviteter (med mye intern kommunikasjon), mens prosesser kan brukes når man trenger mer isolerte delaktiviteter (med mindre intern kommunikasjon).

b) Diskuter hvordan modusskifter (Mode Switching) utføres.

SVAR:

- Adresserom og instruksjonssett skiftes slik at aktiviteter først kan gis tilgang til og deretter kan nektes tilgang til hele maskinens lager og alle maskinens muligheter.

c) Diskuter hvordan I/O-bufring (I/O Buffering) utføres.

SVAR:

- Et mellomlager i OS-et brukes ved innlesing / utskrift til å unngå kjøreblokkering / swapp-blokkering (enkel bufring), og til å utjevne mindre / større hastighetsvariasjoner (multippel bufring).

d) Angi detaljert - med formell kode eller uformell beskrivelse - hvordan ”Frekvensbasert stakk” (Frequency Based Stack)-metoden for I/O-caching virker.

SVAR:

Kombinerer det beste fra LRU og LFU ved å
betrakte Caching-området som en Stakk
med tre Deler hver med Fast størrelse:

* Ny = Kan ikke øke tilsvarende tellere – Kan ikke fjernes fra
* Middels = Kan øke tilsvarende tellere – Kan ikke fjernes fra
* Gammel = Kan øke tilsvarende tellere – Kan fjernes fra

Sider sklir mellom Ny, Middels & Gammel basert på LRU-prinsippet
Sider velges fra Gammel basert på LFU-prinsippet

Oppgave 2: Kontroll av prosesser (Process Synchronization) – 20 %

a) Forklar i hvilke tilfeller en bør bruke henholdsvis programvarebasert (Software Based) prosesstyring og maskinvarebasert (Hardware Based) prosesstyring.

SVAR:

- Maskinvarebasert prosesstyring må brukes på laveste kodenivå (som for svært korte kritiske regioner), mens programvarebasert prosesstyring må brukes på høyere kodenivå (som for lange kritiske regioner eller generell tidsordning).

b) Diskuter hvordan prosesser (Processes) kan kommunisere trygt og effektivt med hverandre.

SVAR:

- Det opprettes et felles lagerområde som alle har adgang til. Det kontrolleres ved gjensidig utelukkelse at prosessenes aktiviteter hverken kan ødelegge for hverandre ved innlegging eller rote for hverandre ved uthenting. Det kontrolleres også at området hverken er helt fullt ved innlegging eller helt tomt ved uthenting.

c) Angi detaljert - med formell kode eller uformell beskrivelse - hvordan ”Oppdage og opprette” (Deadlock Detection)-metoden for vranglåshåndtering virker.

SVAR:

Symboler som benyttes:

Qij: Forespørsels-matrise
Aij: Allokerings-matrise
Pi: Prosess-vektor
Vj: Tilgjengelighets-vektor
Wj: Uttestings-vektor

Prosedyre for å finne syklus:

A: Initier Wj = Vj; j = 1..m
B: Marker Pi: Aij = 0; j = 1..m
C: Alle markert:
    OK (Ferdig)
D: Marker Pi: Wj ≥ Qij; j = 1..m
E: Minst en markert; i = 1..#:
    Wj := Wj + Aij; j = 1..m
    Fortsett i Ledd C
F: Ingen markert:
    Ikke OK (Ferdig)

Alternativer for å fjerne syklus:

* Omstart alle vranglåste
* Delvis omstart alle vranglåste
* Aborter en etter en

* Delvis aborter en etter en

Oppgave 3: Bruk av lager (Memory Management) – 20 %

a) Forklar i hvilke tilfeller en bør bruke henholdsvis fast partisjonering (Fixed Partitioning), enkel sidedeling (Simple Paging) og virtuelt lager med sidedeling (Virtual Memory with Paging).

SVAR:

- Alle tre bør brukes når en ikke ønsker ekstern fragmentering, men aksepterer intern fragmentering. Virtuelt lager med sidedeling passer når en har svært mange eller svært store programmer. Enkel sidedeling passer når en har mange - men ikke for mange, eller store - men ikke for store programmer. Fast partisjonering passer når en har et mindre antall ganske små programmer.

b) Diskuter hvordan side- og segmenttabeller (Page & Segment Tables) kan struktureres slik at de kan aksesseres effektivt og riktig.

SVAR:

- For å unngå oppslag i komplette, men tynne tabeller med kun implisitte sidenr / segmentnr ved et innslags plass i tabellen, bruker en tabeller som utvides med eksplisitte sidenr / segmentnr. Aksess av slike tabeller kan da skje lineært uten noe hjelp, hashet med programvarehjelp, eller assosiativt ved maskinvarehjelp.

c) Angi detaljert - med formell kode eller uformell beskrivelse - hvordan ”4 sjansers klokke” (UM-Clock)-algoritmen virker.

SVAR:

U-bit settes hver gang en side refereres av en prosess
M-bit settes hver gang en side endres av en prosess

Ved ettersøking av en side for utbytting sjekkes alle residerende sider sirkulært, startende etter forrige utbytting, og etter følgende fire sekvensielt ordnete kriterier:

1. (U, M) = (0, 0)
2. (U, M) = (0, 1)
3. (U, M) = (1, 0)
4. (U, M) = (1, 1)

U-bit resettes i runde 2 når en side påtreffes,
og ikke utvelges ved ettersøking for utbytting

Således finnes alltid en side hvor
(U, M) = (0, 0) / (0, 1) senest i runde 4

Oppgave 4: Kjøring av prosesser (Process Scheduling) – 20 %

a) Forklar i hvilke tilfeller en bør bruke henholdsvis prosessavbryting (Process Preemption) og prosessprioritering (Process Prioritization).

SVAR:

- Prosessavbryting må brukes når en ikke kan risikere at en prosess monopoliserer CPU-kraft for lenge, og prosessprioritering må brukes når en kan oppleve at prosesser har veldig ulik betydning i et system.

b) Diskuter hvordan prosesser og tråder kan tidsstyres annerledes på multiprosessorer (Multi CPUs) enn på singelprosessorer (Single CPUs).

SVAR:

- På multiprosessorer blir det ikke så viktig å unngå at en enkelt prosess monopoliserer en CPU, men det blir derimot viktig at en gruppe av tråder kan kjøre på et sett av CPUer samtidig.

c) Angi detaljert - med formell kode eller uformell beskrivelse - hvordan ”Høyeste responsforhold først” (Highest Response Ratio Next)-algoritmen virker.

SVAR:

Neste prosess velges ut fra:

* Maks (W+S) / S

Med følgende størrelser:

* W = Kun ventetid
* S = Total eksekveringstid

Med følgende kategoriseringer:

* Avbrytbarhet: Nei
* Prioritering: Tja – korte og lange

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

a) Forklar i hvilke tilfeller en bør bruke henholdsvis multiprosessorer (Multi CPUs) og distribuerte systemer (Distributed Systems).

SVAR:

- Multiprosessorer kan brukes når en har et lite antall prosessorer og med aktiviteter som trenger å samarbeide forholdsvis tett, mens distribuerte systemer kan brukes når en har et større antall systemer og med aktiviteter som ikke trenger å samarbeide så tett.

b) Diskuter hovedprinsippene i gruppekommunikasjon (Group Communication).

SVAR:

- En må kunne sende meldinger til en utvalgt mengde prosesser / tråder uten at noen som ikke er medlem i denne gruppen får slike meldinger. En må videre kunne sikre seg at en melding enten når fram til alle medlemmer eller ingen medlemmer av gruppen. En må endelig kunne sikre seg at ett sett av meldinger når fram til medlemmene av gruppen i samme rekkefølge.

c) Diskuter hovedkomponentene i CORBA (Common Object Request Broker Architecture).

SVAR:

- CORBA er en mellomvarearkitektur. Den baserer seg på applikasjonsobjekter som aksesserer objekttjenester og felles fasiliteter via en ORB (Object Request Broker). ORBen inkluderer en kjerne som implementerer basismekanismene, og aksesskomponenter som enten effektivt håndterer ferdig oppsatte forbindeler eller endog tillater uforutsette forbindelser. Arkitekturen inkluderer også grensesnittlager for beskrevne klasser og implementasjonslager for registrerte klasser.

d) Angi detaljert - med formell kode eller uformell beskrivelse - hvordan den helt distribuerte algoritmen (altså ikke ringalgoritmen) for gjensidig utelukkelse (Mutual Exclusion) virker.

SVAR:

Eget Inn-ønske:
Send N-1 * Inn-meldinger m/ Lokalt tidsmerke
Vent på N-1 * OK-meldinger

Andres Inn-meldinger:
Ikke ventende: Send OK
Ventende, høyere tidsmerke: Send OK
Ventende, lavere tidsmerke: Sett i Kø
(Like tidsmerker: Identifikator avgjør)

Eget Ut-ønske:
Send $\leq$ N-1 * OK ut fra Kø