7.2 Memory partitioning

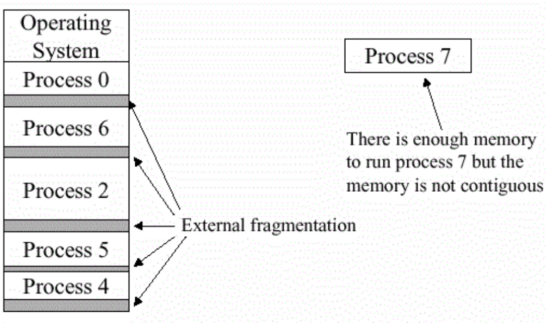
Sunday, April 19, 2020 6:59 PM

Hovedfunksjonen til minnehåndtering er å hente prosesser til hovedminnet for execution hos en prosessor. Minnepartisjonering handler om å dele inn de tilgjengelige delene i hovedminnet til ulike regioner. I de aller fleste moderne multiprogrammerte systemer involverer dette bruk av et virtuelt minne. Det virtuelle minnet er igjen i basert på enten eller begge teknikkene: paging og segmentering. Boken diskuterer ulike teknikker, som både benytter og ikke benytter det virtuelle minnet. Se boken (s. 344+) for mer detaljert beskrivelse.

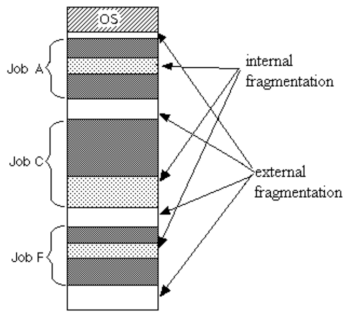
Noen begreper det er kjekt å kunne for videre lesing:

- **Intern fragmentering:** Sløseri av minnet i form av at blokken som lastes inn i minnet ikke dekker hele størrelsen på partisjonen. Det er altså ledig plass i partisjonen som ikke blir utnyttet.
- **Ekstern fragmentering:** Det er totalt nok minne til å kjøre en ny prosess, men det er ikke en stor nok sammenhengende plass for å kjøre prosessen ettersom det er tomme hull i minnet mellom ulike prosesser.
- **Sammenpressing:** En teknikk for å unngå ekstern fragmentering. OS-et bytter om på prosesser så de blir sammenhengende og dermed frigjør nok sammenhengende plass til at en ny prosess kan kjøres.

EKSTERN FRAGMENTERING



INTERN FRAGMENTERING



Minnehåndteringsteknikker

Teknikk	Beskrivelse	Fordeler	Ulemper
Statiske partisjoner	Hovedminnet deles inn i et gitt antall statiske partisjoner ved starten av systemets generering. En prosess kan lastes inn i en partisjon av lik eller ulik størrelse. Se figur 7.2. Se plasseringsalgoritme under.	Enkel å implementere; Lite OS-overhead.	Ineffektiv bruk av minnet på grunn av intern fragmentering; Maksimalt antall aktive prosesser er bestemt/statisk.
Dynamiske partisjoner	Partisjonene lages dynamisk, så hver prosess blir lastet inn i en partisjon av nøyaktig samme størrelse som prosessen. Se plasseringsalgoritme under.	Ingen intern fragmentering; Mer effektiv bruk av hovedminnet.	Ineffektiv bruk av prosessor på grunn av behovet for å sammenpressing for å unngå ekstern fragmentering.
Buddy-system	De to ovennevnte teknikkene har begge ulemper. Statiske partisjoner begrenser antallet aktive prosesser kan utnytte plassen i minnet ineffektivt. Dynamiske partisjoner er mer kompleks å håndtere og har overhead i sammenpressingen. Buddy-systemet er en kombinasjon av disse to teknikkene som raskt tildeler blokker av passende størrelse og samtidig merger nærliggende tomme hull (som ofte oppstår i dynamisk partisjonering). Ved bruk av et tre utnytter buddy-systemet at computere godt håndterer potenser av 2.	Rask til å delegere plass i minnet og frigjøre deler i minnet; Brukes blant annet i Linux.	Sløser med mye minne ved intern fragmentering, siden alle forespørslene er rundet opp til 2-er potens; Det er visstnok 20% som gjennomsnittlig sløses bort (visstnok).
Enkel paging	Hovedminnet deles inn i flere like store frames. Hver prosess deles inn i et antall like store pages av samme lengde som frames. En prosess blir lastet inn ved å laste alle de tilhørende pages inn i tilgjengelig, men ikke nødvendigvis sammenhengende frames.	Ingen ekstern fragmentering.	En liten del intern fragmentering.
Enkel segmentering	Hver prosess deles inn i et antall segmenter. En prosess blir lastet inn ved å laste alle de tilhørende segmentene inn i dynamiske partisjoner som ikke nødvendigvis må være sammenhengende.	Ingen intern fragmentering; Forbedret utnyttelse av minnet og redusert sammenpressingsoverhead sammenlignet med dynamisk partisjonering.	Ekstern fragmentering.
Virtuell minne paging	Samme som for enkel paging, men det er ikke nødvendig å laste inn alle pages tilhørende en prosess. Fraværende pages kan innhentes automatisk senere dersom det er behov.	Ingen ekstern fragmentering; Høyere grad av multiprogrammering; Stort virtuelt adresserom.	Overhead i form av kompleks minnehåndtering.
Virtuell minne segmentering	Samme som for enkel segmentering, men det er ikke nødvendig å laste inn alle segmentene tilhørende en prosess.	Ingen intern fragmentering; Høyere grad av multiprogrammering; Stor	Overhead i form av kompleks minnehåndtering.

segmentering	Fraværende segmenter kan innhentes automatisk senere dersom det er behov.	virtuelt adresserom; Beskyttelse- og delings-støtte.	

Plasseringsalgoritme (Statiske partisjoner)

For blokker med lik størrelse vil en prosess plasseres i hovedminnet, så fremt det er ledig plass. Avhengig av tilstanden til prosessen kan det oppstå swapping.

Ved ulike størrelser kan enten prosessen plasseres i regionen det er minst tilgjengelig plass i hos hovedminnet, men likevel nok plass til prosessen. Dette kan medføre at de største regionene ikke blir brukt hvis ingen prosesser krever stor nok plass. Dermed er det en annen metode hvor prosesser hentes fra en enkel kø og blir valgt ut fra passende størrelse iht. region. Se figur 7.3.

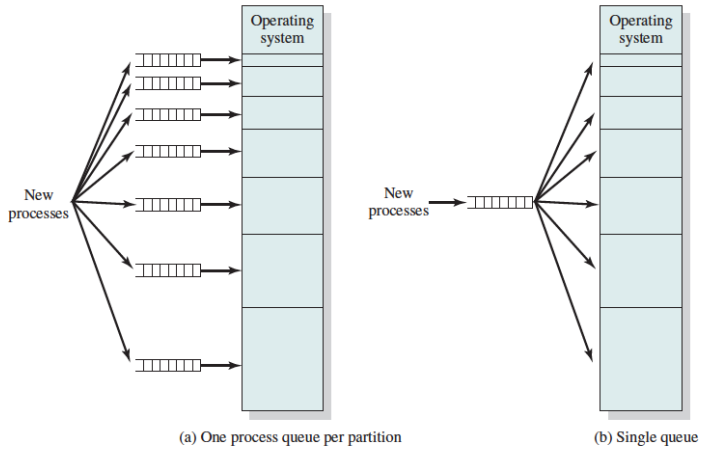


Figure 7.3 Memory Assignment for Fixed Partitioning

Plasseringsalgoritme (Dynamiske partisjoner)

Dynamisk partisjonering har tre ulike metoder:

- **Best-fit:** Velger den blokken som matcher størrelsen til forespørselen best.
- **First-fit:** Scanner minnet fra starten av og velger den første tilgjengelige blokken som er stor nok.
- **Next-fit:** Starter å scanne minnet fra lokasjonen av den siste plasseringen og velger den neste tilgjengelige blokken som er stor nok. Denne metoden er ofte dårligere enn first-fit ettersom den største ledige blokken blir ofte fragmentert.

Relokering

Som forklart tidligere er det ikke uvanlig at OS-et bytter ut prosesser for å frigjøre sammenhengende plass i minnet, i samsvar med sammenpressingsteknikken. Altså er ikke lokasjonene til instruksjonene og dataen som refereres av en prosess alltid på samme sted. For å håndtere denne utfordringen har vi ulike typer for adressene:

- **Logisk adresse:** En virtuell adresse brukeren kan se og som dermed kan brukes som en referanse. Denne genereres av programmet ved kjøretid.
- **Relativ adresse:** En relativ adresse er spesifisert ved å indikere avstanden til en annen adresse (et gitt punkt), for eksempel prosessor-registeret.
- **Fysisk adresse:** Denne adressen refererer til en faktisk lokasjon i hovedminnet. Også kalt absolutt adresse. Bruker kan ikke se denne adressen.

Det hardware-mekanismers oppgave å oversette relative adresser til fysiske adresser.