Vi ser for oss en enkel scheduleringsalgoritme. Prioriteten for en prosess beregnes som forholdet mellom CPU tiden som er brukt av prosessen og virkelig tid som har gått. Jo lavere tall jo høyere prioritet. Prioriteter re-kalkuleres hvert tiende sekund. - Hva slags jobber favoriseres ved denne typen algoritme? - Hvis det ikke utføres noe I/O, vil denne algoritmen reduseres til en round-robin algoritme. Forklar hvorfor. - Diskuter denne algoritmen i forhold til de generelle hensiktene som man ønsker å oppnå med schedulering.

Svar: Hvert tiende sekund får hver prosess en prioritet som er omvendt proporsjonal med brukt kjøretid over de siste 10 sekundene Vi har P = 1/(T/10) hvor P og T er henholdsvis prioritet og brukt kjøretid. De prosessene som har brukt lite kjøretid vil få høy prioritet. Hvis (T == 0) vil prioriteten ikke regnes ut og settes til høyest mulig prioritet.

Hva slags jobber favoriserer denne typen algoritme?

Svar: I/O-intense jobber får veldig høy prioritet, fordi de vanligvis bruker lite kjøretid på CPU. CPU-operasjoner tar ofte lang tid, og prosessene som utfører dem må vente i blocked-state.

Hvis det ikke utføres noe I/O, vil denne algoritmen reduseres til en round-robin algoritme. Forklar hvorfor.

Svar: Alle prosessene vil ha samme prioritet etter hver hele runde, fordi de bruker hele tiden de får på CPU. Merk antagelse om at alle prosessene kjører like lange de 10 sekundene.

Diskuter denne algoritmen i forhold til de generelle hensiktene som man ønsker å oppnå med schedulering.

Svar: Ulempen er at CPU-bundne prosesser kan få “starvation”, lang ventetid siden de må vente på de I/O prosessene. Fordelen er for I/O-drevne prosesser siden de får lite ventetid og dødtid, altså tid det tar å vente mellom prosesser. Dermed god responstid, men kan gi potensielt mye overhead siden vi baserer på prioritet og time-slices. Men det kommer anpå hvilket system vi har å jobbe med. Prioriteres throughput og CPU så er ikke denne algoritmen relativt god.