# Opgave 1

|  |  |  |  |
| --- | --- | --- | --- |
| p0 | p1 |  | turn = 0 |
| Flag[true] |  |  |  |
|  | flag [false] | |  |
| Crit () |  |  |  |
|  | Loop() |  |  |
|  |  |  | turn 1 |
| flag [false] |  |  |  |
|  | flag [true] |  |  |
| Loop() |  |  |  |
|  | Crit () |  |  |
| Flag[true] |  |  |  |
|  | Rem () |  |  |
| Crit () |  |  |  |
| Rem () |  |  |  |

Op dit moment kunnen beide processen in de critical section komen doordat beide processen hun flag[] op true kunnen zetten. Als turn 1 is dan wordt in de loop flag op true gezet terwijl het andere proces nog in de kritieke sectie zit. Dit kan voor meerdere processen werken als turn weer op 0 wordt gezet als een andere proces de critical section in gaat.

# Opgave 2

De firmware (BIOS) laadt een bootloader van de schijf. De bootloader is een klein programma dat tijdens het opstarten de OS kernel in het geheugen laadt en uitvoert.

# Opgave 3

# Opgave 4

## - Trap:

Een trap is een software gegenereerde interrupt die optreedt door een error of een specifiek verzoek van een user program. Bij het uitvoeren van een programma in de usermode kunnen pogingen om toegang te krijgen tot het geheugen van de OS of het geheugen van andere gebruikers resulteren in een trap naar het OS. Het besturingssysteem behandelt deze poging als een fatale fout. Dit beschermingsmechanisme voorkomt dat een gebruikersprogramma per ongeluk of opzettelijk de code of de gegevensstructuren van het OS of andere gebruikers wijzigt.

## - System call

Een interface waarmee applicaties OS-diensten aanroepen, zoals procescontrole, bestandsbeheer of inter-process communication (IPC), is essentieel. Deze interface schakelt van gebruikersmodus naar kernelmodus. Voorbeelden van dergelijke system calls zijn fork(), exec(), read() en write(). Een system call binnen een programma zorgt voor een overgang naar kernelmodus om OS-diensten te gebruiken en vormt de interface tussen gebruikersprogramma’s en de OS-kernel. Meestal worden deze systeemaanroepen aangeboden via high-level API's, zoals POSIX voor UNIX/Linux of Win32 voor Windows.

Voorbeeld: een open() system call om een bestand te openen, of fork() om een nieuw proces te creëren. Bij het uitvoeren van een system call voert het OS namens het proces de gevraagde geprivilegieerde acties uit en retourneert het resultaat of een foutcode.

# Opgave 5

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Proces | p1 | p2 | p3 | p4 |  |  |  |  |  |  |  |  |
| Aankomsttijd | 0 | 2 | 4 | 1 |  |  |  |  |  |  |  |  |
| Actieduur | 4 | 4 | 2 | 2 |  |  |  |  |  |  |  |  |
| Prioriteit | 3 | 2 | 5 | 1 | prio 1 = hoogste 5 = laagste | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| a | p1 | p4 | p2 |  | p3 |  |  |  |  |  |  |  |
| 1. FCFS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|  | p1 | p1 | p1 | **p1** | p4 | **p4** | p2 | p2 | p2 | **p2** | p3 | **p3** |
|  | |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | p1 | p4, p1 | p1, p2, p4 | p2, p4, p1 | p4, p1, p3 p2 | p1, p3, p2 | p3, p2 | p2, p3 | p3,p2 | p2 |  |  |
| 2. RR (1) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|  | p1 | p1 | p4 | p1 | p2 | **p4** | **p1** | p3 | p2 | **p3** | p2 | **p2** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| prio: p4, p2, p1, p3 |  |  |  |  |  |  |  |  |  |  |  |  |
|  | p1 | p4 | p2, p1 |  | p3 |  | p1, p3 |  |  |  |  |  |
| 3. Preemtive priority | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|  | p1 | p4 | **p4** | p2 | p2 | p2 | **p2** | p1 | p1 | **p1** | p3 | **p3** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | p1 | p4 | p2 | p4, p2 | p3 | p2, p3 |  |  |  |  |  |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 4. Non-Preemtive priority | p1 | p1 | p1 | **p1** | p4 | **p4** | p2 | p2 | p2 | **p2** | p3 | p3 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| b. |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. FCFS | Gemiddelde turnarround time: | | | p1 = 4, p2 = 8, p3 = 8, p4 = 5 = (25 / 4 = **6.25**) | | | |  |  |  |  |  |
|  | Gemiddelde wating time: | | | p1 = 0, p2 = 4, p3 = 6, p4 = 3 = (13 / 4 = **3,25**) | | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2. RR (1) | Gemiddelde turnarround time: | | | p1 = 7, p2 = 10, p3 = 6, p4 = 5 = (28 / 4 = **7**) | | | |  |  |  |  |  |
|  | Gemiddelde wating time: | | | p1 = 0, p2 = 2, p3 = 3, p4 = 1 = (6 / 4 = **1.5**) | | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| prio: p4, p2, p1, p3 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3. Preemtive priority | Gemiddelde turnarround time: | | | p1 = 10, p2 = 5, p3 = 8, p4 = 2 = (25 / 4 = **6.25**) | | | |  |  |  |  |  |
|  | Gemiddelde wating time: | | | p1 = 0, p2 = 1, p3 = 8, p4 = 0 = (9/ 4 = **2,25**) | | | |  |  |  |  |  |