Planificació de Processos

Unitat 3 - Nel Banqué Torné

---

1. Introducció a la Planificació

La planificació en sistemes operatius és el procés que determina com els diferents processos accedeixen als recursos del sistema, amb l'objectiu de maximitzar el rendiment i proporcionar una resposta ràpida a les necessitats dels usuaris. El rol del planificador és essencial per mantenir l'eficiència del sistema, equilibrant la càrrega de treball i optimitzant el temps de resposta segons els recursos disponibles.

Objectius Principals

-Millorar el rendiment dels recursos del sistema.

-Prioritzar l'accés als recursos segons la importància i les característiques de cada procés.

-Complir amb els deadlines per assegurar una bona experiència d'usuari.

-Mantenir la CPU ocupada per maximitzar el rendiment del sistema.

El planificador utilitza un algorisme de planificació per prendre decisions i assolir aquests objectius.

---

2. Tipus d'Algorismes de Planificació

2.1 Apropiatius i No Apropiatius

No Apropiatius: Un procés continua executant-se fins que es bloqueja o allibera la CPU de manera voluntària. No es poden fer interrupcions fins que el procés hagi acabat.

Apropiatius: Un procés només pot executar-se durant un temps determinat. Quan aquest temps s'acaba, la CPU retorna al planificador, que decideix si continua amb el mateix procés o passa al següent.

2.2 Planificació a Curt Termini

El **planificador de curt termini** determina quin procés s'executa en cada moment, mentre que el **dispatcher** és l'encarregat de passar la CPU al procés seleccionat.

Funcions del Dispatcher

- Guardar el context del procés sortint.
- Restaurar el context del procés entrant.
- Canviar la CPU a mode d'usuari i saltar al punt d'execució del programa.

Aquest procés ha de ser ràpid per minimitzar els temps de canvi entre processos.

---

3. Algorismes Específics

3.1 Algorisme FIFO i Efecte Convoy

L'**algorisme FIFO** utilitza una cua on els processos s'executen segons l'ordre d'arribada. Això pot causar l'efecte convoy, on els processos que requereixen molta CPU bloquegen aquells que no en necessiten tant, generant temps d'espera llargs.

3.2 Algorisme de Temps Restant Curt (SRT)

Aquest algorisme interromp el procés en execució si arriba un altre procés amb un temps de burst més curt, afavorint així una eficiència global.

---

4. Envelliment en Algorismes de Prioritat

En els algorismes de prioritat, els processos amb baixa prioritat poden quedar en espera indefinidament si arriben processos amb prioritat més alta. Per evitar aquesta situació, s'aplica una tècnica d'envelliment que augmenta la prioritat dels processos que han estat esperant massa temps.

Funció d'Envelliment

```
for each process in processes:
    if process.waitingTime >= agingThreshold:
        process.increasePriority()  # Augmenta la prioritat del procés
```

Això garanteix que cap procés quedi pendent indefinidament i que tots tinguin oportunitat d'executar-se.

---

5. Comparativa d'Algorismes de Planificació

Cada algorisme té punts forts i febles, i el seu rendiment depèn del context. Quan es considera el cost del canvi de context, els resultats poden variar. Per exemple:

- **SJF i FCFS** han tingut dos canvis de context en un escenari de prova.
- **Round Robin**, amb un quantum curt, ha generat deu canvis de context.
- L'**algorisme de Prioritats** ha requerit tres canvis de context.

Aquests resultats mostren que no hi ha un algorisme perfecte per a totes les situacions, i l'elecció de l'algorisme dependrà de les necessitats específiques del sistema i dels seus usuaris.