Sisteme de operare

26 iunie 2010

Timp de lucru: 90 de minute

NOTĂ: toate răspunsurile trebuie justificate

1. Câte procese copil, respectiv părinte poate avea un proces la un moment dat?

Un proces poate avea, la un moment dat, un singur proces părinte și oricâte procese copil, în limita resurselor sistemului. Procesul init poate fi considerat un proces particular care nu are un proces părinte.

2. Un proces execută secvența:

```
for (i = 0; i < 42; i++)
      a++;
```

În timpul execuției secvenței, procesul este preemptat și este planificat alt proces. Dați exemplu de o situație care poate genera preemptarea.

Întrucât secțiunea de mai sus nu este blocantă, procesul poate fi preemptat dacă îi expiră cuanta de timp sau dacă în sistem există un proces cu prioritate superioară pregătit pentru execuție. Această situație este declanșată de apariția întreruperii de ceas.

3. Dați exemplu de o funcție thread safe dar non-reentrantă. Explicați.

O funcție thread safe dar non-reentrantă permite rularea acesteia în context multithreaded dar nu permite existența simultană a două fluxuri de execuție în contextul aceluiași thread/proces. Un exemplu este o funcție care folosește locking. De forma:

```
int my_function(void)
{
    lock(&mutex);
    … /* TODO */ …
    unlock(&mutex);
}
```

4. Se consideră următorul cod:

```
void f()
{
      int *z = malloc(sizeof(int));
      [...]
      printf("z = %p\n", z);
      printf("&z = %p\n", &z);
}
```

După rularea secțiunii se afișează mesajul:

```
      z = 0x12345678
      &z = 0x87654321
```

Asociați adresele z, &z cu secțiunile spațiului de adresă al unui proces: .text, .data, .bss, heap și stack.

z este o variabilă de tip pointer – conținutul acesteia este o adresă. Adresa punctează către o zonă din heap (fiind rezultatul întors de apelul malloc).

&z reprezintă adresa variabilei v. Variabila este o variabilă locală unei funcții deci este alocată pe stivă.

Avem o variabilă alocată pe stivă (adresa ei este o adresă din stivă), iar conținutul acelei variabile (pointer) este o adresă întoarsă de apelulul malloc, adică o adresă din heap.

5. Explicați modul în care se poate produce starvation pe un sistem cu planificare SRTF (Shortest Remaining Time First).

Dacă în cadrul sistemului apar în coada ready procese cu timp de rulare redus, acestea vor fi planificate primele. Presupunând un flux continuu de procese cu timp de rulare redus, procesele cu timp de rulare mare vor ajunge să se execute foarte rar sau deloc, adică să se producă fenomenul de starvation.

6. După schimbarea contextului între două thread-uri, care clase registre au valori diferite (înainte și după schimbarea de context): registrele generale, registrul de stivă, registrele de segment.

La schimbarea de context între două thread-uri, majoritatea registrelor se schimbă. Fiecare thread dispune de valori proprii ale registrelor. Astfel, registrele generale și registrul de stivă se schimbă. Sistemele de operare moderne folosesc rar registrele de segment, astfel că în general acestea nu vor fi schimbate.

În plus, anumite registre interne procesorului, inaccesibile din user space (ring3 pe o arhitectură x86) își pot păstră valorile în cazul thread-urilor diferite (registre precum cr2, cr3 pe o arhitectură x86).

7. De ce anumite zone din bibliotecile partajate sunt mapate read-write? Dati un exemplu.

Bibliotecile partajate conține zone r-x (cod), r-- (read-only data) și rw- (date). Zonele read-write conțin variabile care pot fi scrise de procesul ce folosește biblioteca, precum variabila errno în cazul bibliotecii standard C.

8. Exceptând apelurile de sistem, dați exemplu de situație în care procesorul comută în kernel space.

Procesorul execută cod kernel în cazul unor solicitări din user space (apel de sistem) sau de la hardware (întreruperi). Procesorul execută, astfel, cod kernel, exceptând apelurile de sistem, în momentul sosirii unei întreruperi.

9. Un sistem dispune de N procese. Fiecare proces dispune de M pagini virtuale nealocate. Sistemul dispune de o singură pagină fizică disponibilă. Care este numărul maxim de pagini virtuale care pot fi asociate cu pagina fizică?

*Oricâte pagini virtuale pot fi asociate cu o pagină fizică. Implementări de tipul mmap permit maparea unei zone de memorie virtuale peste o zonă de memorie fizică. În situația de mai sus, un proces poate mapa toate cele M pagini virtuale proprii peste acea pagină fizică. În total, pentru cele N procese, se pot mapa N*M pagini virtuale.*

10. Explicați de ce nu se poate implementa mecanismul de swapping pe un procesor fără unitate de management al memoriei.

Unitatea de management a memoriei (MMU) este responsabilă cu translatarea paginilor virtuale în pagini fizice. Absența MMU conduce la absența mecanismului de memorie virtuală. Mecanismul de memorie virtuală permite existența unui spațiu virtual de adrese care depășește spațiul fizic – spațiul suplimentar poate fi furnizat de disc, prin intermediul swap-ului. În cazul absenței mecanismului de memorie virtuală, nu se poate referi/folosi spațiul de swap, deci nu se poate implementa mecanismul de swapping (evacuare pe disc și recuperarea paginilor de pe disc).

11. Un inode dispune de 10 pointeri de indirectare simplă a blocurilor de date. Un bloc ocupă 4096 de octeți. Știind că un dentry ocupă 64 de octeți, câte intrări poate avea maxim un director?

10 pointeri de indirectare simplă punctează către blocuri care conțin, la rândul lor, pointeri. Considerând că un pointer ocupă 4 de octeți, rezultă că un bloc de pointeri conține 4096/4 = 1024 de pointeri.

*Cei 10 pointeri de indirectare simplă vor referi 10 blocuri care conțin, la rândul lor, 10*1024 pointeri adică 10240.*

*Fiecare dintre cei 10240 pointeri punctează către un bloc de date. Fiecare bloc de date ocupă 4K. Rezultă așadar, că un inode poate referi 10240*4KB de date.*

*În cazul unui director, datele sale sunt un vector (array) de dentry-uri. Numărul maxim de dentry-uri se obține împărțind spațiul maxim ce poate fi referit de un inode la dimensiunea unui dentry. În consecință, un director poate conține 10240*4KB / 64 dentry-uri, adică 10240 * 64 = 655360.*