Nume și grupă:

Sisteme de Operare

4 septembrie 2014

Timp de lucru: 60 de minute

Notă: Toate răspunsurile trebuie justificate

1. (**7 puncte**) Care este un avantaj al folosirii tabelei de pagini ierarhice față de tabela de pagini simplă?

2. (**7 puncte**) De ce un sistem este cu atât mai încărcat cu cât numărul de procese din cozile `READY` crește (adică mai multe procese în cozile `READY` înseamnă încărcare mai mare)?

3. (**7 puncte**) În ce situație o operație de tip `lock()` pe un mutex blochează thread-ul curent și în ce situație nu îl blochează?

4. (**7 puncte**) De ce au dispozitivele de tip bloc nevoie de operații mai rapide (cu throughput mai mare) decât dispozitivele de tip caracter?

5. (**7 puncte**) Cu ce diferă un apel de bibliotecă de un apel de sistem?

6. (**10 puncte**) În ce situație se poate rula cod pe stivă, chiar în cazul folosirii unui mecanism de *stack smashing protection* (*canary value*)?

7. (**10 puncte**) La montarea unui sistem de fișiere se poate folosi opțiunea `noatime`. Opțiunea înseamnă că **nu** va fi actualizat câmpul `atime` (timestamp de acces) al unui inode în momentul accesării (citirii sau scrierii inode-ului). De ce este avantajoasă această opțiune în cadrul unui sistem de fișiere încărcat (cu accese dese la fișiere)?

8. (**10 puncte**) Fie secvența de instrucțiuni de mai jos:

```
*a = 42;      /* first dereferencing */
sleep(5);     /* sleep for 5 seconds */
*a = 42;      /* second dereferencing */
```

unde `a` este un pointer la un întreg (`int *`). Dați exemplu de situație în care prima dereferențiere **nu** cauzează page fault, dar a doua dereferențiere cauzează page fault.

9. (**10 punct**) Un proces în Linux are, în mod obișnuit, tabela de descriptori de fișiere limitată la 1024 de intrări. De ce în cazul unui server TCP încărcat, care primește multe conexiuni, este nevoie de creșterea acestei limite?

10. (**10 puncte**) De ce este mai probabilă apariția unui *stack overflow* în cazul unui proces multi-threaded față de un proces single-threaded? (*stack overflow* = depășirea limitei stivei în cadrul spațiului de adrese al unui proces)

11. (**15 puncte**) Ne propunem implementarea unui framework de messaging (message queue framework). Cu ajutorul acestui framework, aplicații diferite pot comunica unele cu celelalte. Există patru concepte importante: Publishers (cei care produc mesaje), Consumers (cei care consumă mesaje din cozi), Exchanges (cei care primesc mesajele în cozi de la Publishers și apoi le transmit către Consumers), Queues (cozi de mesaje, create de Consumers și care stochează mesajele). Framework-ul trebuie să asigure scalabilitate și performanță. În general, Consumers, Exchanges și Publishers se găsesc pe sisteme fizic diferite; sunt conectați prin Internet/rețea.

Definiți, la nivel de pseudocod, metodele framework-ului folosite de Publishers și Consumers.

Ce probleme de scalabilitate pot apărea la nivelul framework-ului? (adică de la un nivel în sus se vor resimți probleme de performanță)

Ce soluții de rezolvare a problemelor de scalabilitate există? (atât la nivelul framework-ului, cât și la nivelul infrastrcturii folosite în instalare)

Cum asigurați o performanță ridicată a framework-ului și pentru o utilizare simplă (fără probleme de scalabilitate)?

În conformitate cu ghidul de etică al Departamentului de Calculatoare, declar că nu am copiat și nu voi copia la această lucrare. De asemenea, nu am ajutat și nu voi ajuta pe nimeni să copieze la această lucrare.

Nume și grupă:

Semnătură:.....