Nume și grupă:

Sisteme de Operare

8 septembrie 2017

Timp de lucru: 90 de minute

Notă: Toate răspunsurile trebuie justificate

1. (**7 puncte**) Ce se întâmplă cu cursorul de fișier dacă un proces deschide același fișier de două ori?

2. (**7 puncte**) În Linux, putem porni un proces nou cu apelul `fork()`, sau cu apelul `fork()` `+ exec()`. Care variantă este mai rapidă? (explicați răspunsul)

3. (**7 puncte**) În secvența de mai jos, explicați când va aloca sistemul de operare memorie pentru procesul copil:

```
int i = 0;
if (fork() == 0) {
        i++;
}
```

4. (**7 puncte**) Ce mecanism IPC are *overhead* zero la transferul datelor, indifirent de dimensiune? Explicați răspunsul.

5. (**7 puncte**) Majoritatea apelurilor de sistem necesită tranziția în mod nucleu. De ce sunt aceste tranziții necesare?

6. (**10 puncte**) Cum se explică faptul că două sisteme Linux conectate cu un link de `10Gbps` pot transfera UDP la o viteză de maximum `5Gbps` și TCP la `10Gbps`?

7. (**10 puncte**) Pe procesoarele x86 pe 64 biți multicore, scrierea unei valori de 64 de biți este atomică. Explicați cum putem folosi această proprietate pentru a implementa un buffer circular (cu un singur producător și un singur consumator) fără lock-uri.

8. (**10 puncte**) Descrieți o modificare care ar putea fi implementată în gcc (compilator) pentru a preveni atacurile de tip `buffer overflow`.

9. (**10 puncte**) De cele mai multe ori, la a doua citire dintr-un fișier, cititul datelor durează mult mai puțin decât la prima citire. De ce se întâmplă acest lucru?

10. (**10 puncte**) Un programator scrie un buffer de `10000` caractere pe ecran, câte un caracter per scriere, comparând performanța `write()` și `printf()`. Care variantă este mai rapidă? Este posibil ca cele două variante sa ofere timp de execuție similar?

11. (**25 puncte**) Un programator observă că multe blocuri din fișiere diferite au conținut de date identic. Pentru a reduce spațiul utilizat pe disc, dorește să implementeze deduplicare: un mecanism care să permită reducerea spațiului ocupat de fișiere cu conținut similar, pe disc. Ajutați programatorul să definească API-ul pentru apelul `duplicate_file()`, un apel de sistem similar cu `fork()` dar care se aplică unui fișier pentru a crea o replică identică care nu alocă nici un bloc nou pe disc. Dupa aceea, cele două fișiere duplicate vor avea în comun blocurile identice, și vor avea o copie proprie a blocurilor care diferă (i.e. comparabil cu mecanismul *copy on write*).

a. Explicați ce parametri sunt necesari pentru `duplicate_file()`, ce valoare întoarce, și care este semnificația valorilor întoarse. (**3 puncte**)
b. Explicați ce modificări sunt necesare în inode pentru a implementa apelul `duplicate_file()`. (**4 puncte**)
c. Scrieți (pseudo-)cod care implementează apelul `duplicate_file()`. (**10 puncte**)
d. Scrieți (pseudo-)cod care implementează apelurile `read()` și `write()` care funcționează cu `duplicate_file()`. (**8 puncte**)

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:.....