Nume și grupă:

Sisteme de Operare

10 iunie 2011

Timp de lucru: 60 de minute

Notă: Toate răspunsurile trebuie justificate

1. Un sistem pe 32 de biți folosește memorie virtuală și pagini de 4K. Care dintre următoarele adrese virtuale pot referi, la un moment dat, aceeași adresă fizică:

- 0x43210078
- 0x65430278
- 0x76540018
- 0x54320078
- 0x87650021

2. Cum explicați faptul că, deși, în general, procesele daemon au ca părinte procesul init, nu toate au fost create de init?

3. De ce mecanismul ASLR (Address Space Layout Randomization) are sens pe sistemele pe 64 de biți, dar relevanță scăzută pe sistemele pe 32 de biți?

4. De ce nu au sens algoritmi de planificare de operații pe disc precum C-SCAN, C-LOOK, pe discuri flash?

5. În ce situație, instrucțiunea

```
*a = 42;
```

generează o operație de swap out?

6. Un proces execută următoarea secvență de cod:

```
fd1 = open(\a.txt", O_RDWR);
write(fd1, \anaaremere", 10);
Se execută, ulterior, următoarea secvență în diferite contexte:
fd2 = open(\a.txt", O_RDWR);
write(fd2, \bogdanarepere", 13);
```

Contextele sunt:

- în cadrul aceluiași proces
- într-un thread nou
- într-un proces copil al procesului inițial
- într-un proces diferit și nelegat de procesul inițial

Fișierul `a.txt` va avea, în final, același conținut, indiferent de contexte. Justificați. Se presupune că toate apelurile reușesc.

7. Dați exemplu de situație în care un apel `read` sincron blocant nu se blochează (apelul reușește - se întoarce cu succes).

8. Inspectarea spațiului de adresă al unui proces arată că dimensiunea stivei la crearea procesului este de 128KB. Cu toate acestea, pe durata execuției procesului se apelează o funcție care folosește o variabilă locală ce are dimensiunea de 8MB fără a cauza o eroare. Cum explicați acest lucru?

9. Fie următoarea secevență de cod:

```
#include <stdio.h>

void f(void)
{
        printf("hello\n");
}

int main(void)
{
        int *ptr;

        ptr = (int *) f;
        printf("0x%08x\n", *ptr);

        *ptr = 0x12345678;

        return 0;
}
```

La rularea codului se obține rezultatul:

```
---
0xe5894855
Segmentation fault
---
```

Cum explicați?

10. De ce, pentru o implementare de thread-uri la nivelul nucleului (kernel-level threads) durata schimbării de context între două thread-uri din procese diferite este vizibil mai mare decât durata schimbării de context între două thread-uri ale aceluiași proces?

În conformitate cu ghidul de etică al Catedrei 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:.....