Nume și grupă:

Sisteme de Operare

5 septembrie 2011

Timp de lucru: 60 de minute

Notă: Toate răspunsurile trebuie justificate

1. Într-un sistem de fișiere, numele fișierului este reținut în inode. Poate conține sistemul de fișiere link-uri simbolice (symlinks)? Dar hard links?

2. O structură de date este accesată din contextul unui handler de tratare a unui semnal și din contextul normal de rulare a unui program. Cum se asigură sincronizarea accesului la acea structură?

3. Un proces deține mai multe thread-uri. Se detectează că un thread a suferit un atac de tipul stack overflow în timpul rulării sale. Este suficientă terminarea thread-ului pentru a garanta încheierea atacului?

4. Fie programul de mai jos:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
        long *ptr;

        ptr = (long *) main;

        printf("*ptr = %x\n", *ptr);
        *ptr = 0x1234;

        return 0;
}
```

La rularea sa se obține:

```
$ ./exec
*ptr = e5894855
Segmentation fault
```

Cum explicați?

5. Fie cele două instrucțiuni de mai jos. Presupunând că nu există erori, în ce situație instrucțiunea (A) durează mai mult decât (B) și invers?

- (A) `*ptr = 0x1234;`
- (B) `read(fd, buffer, 1024);`

6. În implementarea uzuală a unui nucleu sistem de operare, un pipe este reprezentat de un buffer. Dați câte un exemplu de situație în care, primind ca argument un descriptor de pipe, apelul `read()`, respectiv `write()` se blochează.

7. Un proces conține trei thread-uri. Unul dintre thread-uri realizează un access nevalid la memorie. Ce se întâmplă cu celelalte două thread-uri?

8. Trei thread-uri accesează o aceeași structură de date: unul dintre thread-uri accesează structura în mod read-write, iar celelalte thread-uri în mod read-only. Ce thread trebuie să folosească primitive de sincronizare?

9. De ce este mai facil de realizat limitarea spațiului de disc pentru o mașină virtuală VMware decât pentru un container OpenVZ?

10. Fie trei procese A, B, C aflate în relație de părinte - copil: A este proces părinte pentru B, iar B este proces părinte pentru C. Cronologic, au loc următoarele acțiuni (`t2 > t1`):

- `t1`: C se termină, fără a fi așteptat de B
- `t2`: B se termină, fără a fi așteptat de A
- după `t2`: A rulează în continuare, fără a-l aștepta pe B

Începând cu momentul `t1` utilizatorul investighează starea proceselor și observă, în fiecare moment, un singur proces în starea zombie. De ce?

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:.....