Sisteme de operare

26 iunie 2009

Timp de lucru: 70 de minute

NOTĂ: toate răspunsurile trebuie **justificate**

1. Știind că operațiile de lucru cu pipe-uri sunt atomice, implementați în pseudocod un mutex cu ajutorul pipe-urilor.

2. Durata unei schimbări de context este de 1ms iar overhead-ul unui apel de sistem de 100µs. Totuși, la un moment dat, apelul down(&sem); durează doar 1µs. Apelul se realizează cu succes. Care este explicația?

3. De ce este mai avantajos ca, pe un sistem uniprocesor, după un apel *fork* să fie planificat primul procesul fiu?

4. Există vreo diferență între implementarea simbolului *errno* în contextul unui proces single-threaded față de un proces multi-threaded? Argumentați.

5. Un sistem uniprocesor (single-core) dispune de 64KB L1 cache, 512KB L2 cache și un TLB cu 256 intrări. Pe un sistem de operare cu suport în kernel pentru thread-uri, ce durează mai mult: schimbarea de context între două thread-uri sau între două procese?

6. Comanda pmap afișează informații despre spațiul de adrese al unui proces. În urma rulării de mai jos a comenzii *pmap* se observă următoarele informații despre biblioteca standard C:

```
# pmap 1
base address          size    rights     name
[...]
00007f8c480e6000    1320K    r-x--      libc-2.7.so
00007f8c4842f000      12K    r----      libc-2.7.so
00007f8c48432000       8K    rw---      libc-2.7.so
```

Presupunând că în sistem rulează 50 de procese care folosesc biblioteca standard C, care este spațiul total de memorie RAM ocupat de bibliotecă?

7. Descrieți și explicați în pseudo-asamblare cum acționează suportul de SSP (Stack Smashing Protection) pe un sistem în care stiva crește în sus (de la adrese mici la adrese mari).

8. Pe un sistem de fișiere dat un dentry are următoarea structură:
- 1 octet - lungimea numelui
- 251 octeți – numele
- 4 octeți - numărul inode-ului

O instanță a unui astfel de sistem de fișiere deține un director rădăcină, 5 subdirectoare, iar fiecare subdirector conține 5 fișiere. Câte dentry-uri deține sistemul de fișiere?

9. Un sistem pe 64 de biți folosește pagini de 8KB și 43 de biți pentru adresare într-o schemă de adresare ierarhică pe trei niveluri cu împărțirea (10 + 10 + 10 + 13). Un proces care rulează în cadrul acestui sistem are, la un moment dat, următoarea componență a spațiului de adrese (se începe de la adresa 0):

```
[ text ]             - 16 pagini
[ data ]             - 8 pagini
[ spațiu nealocat ]  - 8168 pagini
[ stivă]             - 8 pagini
```

Știind că o intrare în tabela de pagini ocupă 64 de biți, câte pagini ocupă tabelele de pagini pentru procesul dat?

10. Dați exemplu de situație în care, pentru comunicația cu dispozitivele de I/E, se preferă folosirea polling în loc de întreruperi.

11. Un program execută secvența de cod din coloana din stânga tabelului de mai jos. În coloana din dreapta este prezentat rezultatul rulării programului:

```c
/* init array to 2, 0, 0, 0 ... */
static int data1[1024*1024] = {2, };

static void print_time(char *msg)
// ...

static void init_array(int *a, size_t len)
{
    size_t i;

    for (i = 0; i < len; i += 1024)
        a[i] = 2009;
}

int main(void)
{
    int *data2 = malloc(1024*1024 * sizeof(int));

    print_time("before init data1");
    init_array(data1, 1024*1024);
    print_time("after  init data1");

    print_time("before init data2");
    init_array(data2, 1024*1024);
    print_time("after  init data2");

    return 0;
}
```

```
before init data1: 1245582962s, 753431us
after  init data1: 1245582962s, 767496us

before init data2: 1245582962s, 767524us
after  init data2: 1245582962s, 776012us



---
Se observa ca:
-   durata initializare data1 - 14065us
-   durata initializare data2 -  8488us
```

Cum explicați faptul că inițializarea vectorului *data1* durează mai mult decât inițializarea vectorului *data2*?

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

Nume:..... Grupă: .....

Semnătură: .....