Sisteme de operare

6 septembrie 2009

Timp de lucru: 70 de minute

NOTĂ: toate răspunsurile trebuie justificate

1. Un sistem dispune de următoarele caracteristici
- magistrala de date pe 64 de biți
- overhead-ul impus de un page fault este de 1ms
- nu dispune de memorie cache
- durata unei operații cu memoria este de 100ns

Pe sistem rulează un sistem de operare în cadrul căruia biblioteca standard C folosește un buffer intern de 64K la nivelul fiecărui handle de fișier.
Care din operațiile marcate aldin (bold) de mai jos durează mai mult?

`char buf[32*1024];`
`f = fopen("a.dat", "wb");`
`/* fill buffer */`
`…`
`fwrite(f, buf, 32*1024);`
`fflush(f);`
`fwrite(f, buf, 32*1024);`
	`char buf[32*1024];`
	`int fd;`
	`char *a;`
	`fd = open("a.dat", O_RDWR | O_CREAT | O_TRUNC, 0644);`
	`/* fill buffer */`
	`…`
	`write(fd, buf, 32*1024);`
	`a = mmap(NULL, 32*1024, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);`
	`memcpy(a, buf, 32*1024);`

2. În cadrul problemei celor 5 filozofi se folosește următoarea implementare (în pseudo-C) a funcției eat():

```
mutex_t global_mutex;

void eat(int fork1, int fork2)
{
        lock(global_mutex);
        take_fork(fork1);
        take_fork(fork2);
        do_eat();
        put_fork(fork1);
        put_fork(fork1);
        put_fork(fork2);
        unlock(global_mutex);
}
```

Care este neajunsul acestei implementări?

3. Pe un sistem rulează 50 de procese. La un moment dat este pornită o mașina virtuală VMware Server pe care rulează 50 de procese. Câte procese vor rula pe sistemul gazdă? Dar în cazul pornirii unei mașini virtuale OpenVZ pe care rulează 50 de procese?

4. Pe un sistem de fișiere MINIX se execută operațiile

```
lseek(fd, SEEK_SET, 32*1024);
write(fd, buffer, 1024);
```

Descrieți operațiile asociate asupra structurilor sistemului de fișiere (inode, bitfield, data block, dentry etc.)

5. Un sistem folosește un planificator round-robin non-preemptiv. În cadrul sistemului rulează 5 procese care execută următoarea secvență:
`[ 10ms rulare | 10ms așteptare | 10ms rulare ]`
Cât durează planificarea celor 5 procese?

6. Cum se modifică spațiul de adresă al unui proces la schimbarea de context între două thread-uri?

7. Are sens folosirea operațiilor asincrone în locul celor sincrone în situația de mai jos (pseudo-cod)?

```
AIO_TYPE aioArray[32];
InitializeAsyncIoArray(aioArray);
for (i = 0; i < 32; i++)
        StartAsyncIo(aioArray[i]);
WaitForAllObjects(aioArray);
```

8. Descrieți o situație în care operația:
`memcpy(a, “12345678901234567890”, 20);`
durează mai mult, respectiv mai puțin, decât operația
`getpid();`
Operația epoll_ctl_add rezultă într-un apel de sistem. Overhead-ul unui page fault este de 5ms, iar a unui apel de sistem de 7ms.

9. Pe o arhitectură x86, care registre generale (eax, ebx, ecx, edx, esi, edi, ebp, esp) sunt schimbate în cazul unei schimbări de context între două thread-uri? Dar în cazul unei schimbări de context între două procese?

10. Se presupune că se implementează la nivel hardware o tehnică ce împiedică accesarea zonelor de memorie nealocate la nivel de octet. Cum poate fi folosită această tehnică pentru prevenirea atacurilor de tip buffer overflow la nivelul stivei?

11. Într-un sistem de operare, **4** (patru) procese execută operațiile

```
f1 = open(“a.txt”, O_RDONLY);
f2 = open(“b.txt”, O_RDWR);
a1 = mmap(NULL, 4*1024, PROT_READ, MAP_SHARED, fd1, 0);
a2 = mmap(NULL, 4*1024, PROT_READ | PROT_WRITE, MAP_SHARED, fd2, 0);
printf(“%c”, *a1);
*a2 = ‘a’;
```

Câte pagini de memorie fizică, respectiv virtuală vor fi ocupate în urma operațiilor de mai sus?

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: .....