Nume și grupă:

Sisteme de Operare

8 iunie 2016

Timp de lucru: 90 de minute

Notă: Toate răspunsurile trebuie justificate

1. (**7 puncte**) Precizați un apel care modifică cursorul de fișier și un apel care modifică dimensiunea unui fișier.

2. (**7 puncte**) Precizați și justificați o metrică pentru nivelul de interactivitate al proceselor pentru un planificator.

3. (**7 puncte**) Fie secțiunea de cod de mai jos:

```
char arr[128];
...
arr[140] = '\0';
```

În ce situație instrucțiunea de atribuire rezultă în eroare de acces la memorie (de tip *segmentation fault*) și în ce situație nu?

4. (**7 puncte**) Apelul `malloc()` alocă memorie, în vreme ce apelul `calloc()` o și *zeroizează* (umple spațiul alocat cu valori de zero). De ce în cazul apelului `calloc()` spațiul de memorie fizică/rezidentă (*resident set size*) a procesului crește, dar în cazul `malloc()` nu (sau foarte puțin)?

5. (**7 puncte**) Pe un sistem de operare dat, stiva unui thread este limitată la 8MB. Care este un avantaj și un dezavantaj al creșterii acestei limite (de exemplu la 32MB)?

6. (**10 puncte**) Un sistem pe 64 de biți are suport pentru DEP (*Data Execution Prevention*) și ASLR (*Address Space Layout Randomization*). Presupunând că identificăm într-un program o vulnerabilitate de tip buffer overflow, ce acțiuni urmărim pentru a putea obține un shell?

7. (**10 puncte**) Precizați un scenariu în care două fișiere (inode-uri) diferite ajung să refere același bloc de date.

8. (**10 puncte**) Un buffer circular este un buffer pentru care operațiile ajunse la sfârșitul său, continuă de la început (dacă ai ajuns la ultimul element al buffer-ului, continuă de la primul element). Buffer-ul are două capete (doi indecși): `in` și `out`. Un consumator citește de la `out`, iar un procucător scrie la `in`. Precizați în pseudocod o implementare sincronizată a funcțiilor `consume()` și `produce(item)` folosind un buffer circular.

9. (**10 puncte**) Pentru un program/executabil putem folosi funcționalitatea de `setuid`pentru acțiuni privilegiate. Întrucât această funcționalitate este de tip totul sau nimic (obții întregul set de privilegii ale utilizatorului administrativ), se recomandă folosirea de capabilități. O capabilitate este avantajoasă pentru că permite doar un anumit tip de acțiune privilegiată. De ce nu are sens să existe o capabilitate pentru acțiuni privilegiate cu sistemul de fișiere (adică scrierea și citirea oricărui fișier)?

10. (**10 puncte**) *Load average* este o metrică care este proporțională cu numărul de procese aflate în starea `READY`. De ce *load average*-ul este mai mare pentru un sistem cu mai multe procese de tip *CPU intensive*?

11. (**25 puncte**) Pentru un tip de aplicații dorim să folosim un alocator personalizat de memorie. Știm că acest tip de aplicații alocă, de cele mai multe ori, obiecte de diferite tipuri, fiecare tip ocupând memorie de dimensiune dată (adică chunk-uri de dimensiune dată). Adică se fac arareori alocări de dimensiune variabilă. Detaliați modul de proiectare și implementare a acestei biblioteci. Biblioteca expune API-ul clasic de lucru cu memoria: `malloc()`, `calloc()`, `realloc()`, `free()`.

a. Cum veți gestiona, la nivelul bibliotecii, spațiile libere și spațiile (chunk-urile) ocupate? Precizați ce tip de structuri de date veți folosi și argumentele pentru folosirea spațiilor libere/ocupate. (**7 puncte**)
b. Cum veți asigura o eficiență bună a implementării bibliotecii astfel încât alocarea și dezalocarea să aibă overhead redus? (**6 puncte**)
c. Indicați ce se întâmplă la nivelul bibliotecii și structurilor acesteia în cazul apelurilor `malloc()` și `free()`? (7 puncte)
d. Cum veți oferi suport multithreaded la nivelul bibliotecii, urmărind și asigurarea unei bune eficiențe? (**5 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:.....