Systemy operacyjne

Lista zadań nr 6

Na zajęcia 19 i 25 listopada 2025

Należy przygotować się do zajęć czytając następujące materiały: [1, rozdziały 3.13, 4.4-4.5, 8.11], [2, rozdziały 9, 13, 38, 39].

UWAGA! W trakcie prezentacji należy być gotowym do zdefiniowania pojęć oznaczonych **wytłuszczoną** czcionką.

Zadanie 1. W każdym z poniższych przypadków zakładamy, że początkowa **tożsamość** naszego procesu to: `ruid=1000`, `euid=0`, `suid=0`. Jak zmieni się tożsamość procesu po wywołaniu następujących funkcji: (a) `setuid(2000)`, (b) `setreuid(-1, 2000)`, (c) `seteuid(2000)`, (d) `setresuid(-1, 2000, 3000)`. Odpowiedź uzasadnij posługując się podręcznikami systemowymi `setuid(2)`, `setreuid(2)`, `setresuid(2)`.

Czy proces z tożsamością `ruid=0`, `euid=1000`, `suid=1000` jest uprzywilejowany? Odpowiedź uzasadnij.

Zadanie 2. Jaką rolę pełnią bity uprawnień «`rwx`» dla katalogów w systemach uniksowych? Opisz znaczenie bitów «`set-gid`» i «`sticky`» dla katalogów. Napisz w pseudokodzie i zreferuj procedurę «`bool my_access(struct stat *sb, int mode)`». Pierwszy i drugi argument opisano odpowiednio w `stat(2)` i `access(2)`. Dla procesu o tożsamości zadanej przez `getuid(2)` i `getgroups(2)` procedura «`my_access`» sprawdza czy proces ma **upoważniony** dostęp «`mode`» do pliku o metadanych wczytanych do «`sb`».

Wskazówka: Rozważ uprawnienia katalogu «`/usr/local`» i «`/tmp`».

Zadanie 3. Właścicielem pliku programu `su(1)` jest «`root`», a plik ma ustawiony bit «`set-uid`». Jaką tożsamość będzie miał na początku proces wykonujący «`su`», jeśli przed `execve(2)` było `euid=1000`?

Zreferuj działanie uproszczonej wersji programu `su`[1] zakładając, że wszystkie wywołania systemowe kończą się bez błędów, a użytkownik zdołał się **uwierzytelnić**. Skoncentruj się na funkcjach czytających bazę danych użytkowników, odczytujących i sprawdzających hasło, oraz zmieniających tożsamość procesu.

Zadanie 4. Na podstawie §38.2 i §38.3 wyjaśnij czemu programy uprzywilejowane należy projektować w taki sposób, by operowały z najmniejszym możliwym zestawem upoważnień (ang. *the least privilege*). Zreferuj wytyczne dotyczące projektowania takich programów. Zapoznaj się z §39.1 i wytłumacz czemu standardowy zestaw funkcji systemu uniksowego do implementacji programów uprzywilejowanych jest niewystarczający. Jak starają się to naprawić zdolności (ang. *capabilities*)? Dla nieuprzywilejowanego procesu posiadającego zdolności «`CAP_DAC_READ_SEARCH`» i «`CAP_KILL`» jądro pomija sprawdzanie upoważnień do wykonywania pewnych akcji – wymień je. Kiedy proces użytkownika może wysłać sygnał do innego procesu?

Zadanie 5. Jakie zadania pełni procedura `exit(3)` z biblioteki standardowej? Opisz problemy z buforowaniem plików, które mogą wystąpić dla strumieni biblioteki `stdio(3)` w przypadku użycia wywołań `fork(2)`, `execve(2)` i `_exit(2)`. Jak zapobiec tym problemom? Jaka jest domyślna strategia buforowania strumienia związanego z (a) plikiem terminala (b) plikiem zwykłym (c) standardowym wyjściem błędów «`stderr`».

Piszesz program który używa biblioteki «`stdio`». Działanie programu da się przerwać sygnałem «`SIGINT`». Ma on wtedy opróżnić wszystkie bufory otwartych strumieni i dopiero wtedy wyjść. Zaproponuj rozwiązanie pamiętając, że w procedurach obsługi sygnału nie wolno korzystać z funkcji, które nie są wielobieżne.

[1] `https://git.suckless.org/ubase/file/su.c.html`

Ściągnij ze strony przedmiotu archiwum «so21_lista_6.tar.gz», następnie rozpakuj i zapoznaj się z dostarczonymi plikami.

UWAGA! Można modyfikować tylko te fragmenty programów, które zostały oznaczone w komentarzu napisem «TODO».

Zadanie 6 (2). Program «writeperf» służy do testowania wydajności operacji zapisu do pliku. Nasz microbenchmark[2] wczytuje z linii poleceń opcje i argumenty opisane dalej. Na standardowe wyjście drukuje t trójkątów (opcja «-t») prostokątnych o boku złożonym z l znaków gwiazdki «*» (opcja «-l»). Jeśli standardowe wyjście zostało przekierowane do pliku oraz została podana opcja «-s», to przed zakończeniem programu bufory pliku zostaną zsynchronizowane z dyskiem wywołaniem fsync(2).

Program realizuje pięć wariantów zapisu do pliku:

- Każdą linię trójkąta zapisuje osobno wywołaniem write(2) (argument «write»).
- Używa strumienia biblioteki stdio bez buforowania (argument «fwrite»), z **buforowaniem liniami** (argument «fwrite-line») i **buforowaniem pełnym** (argument «fwrite-full»).
- Wykorzystuje wywołanie systemowe writev(2) do zapisania do «IOV_MAX» linii na raz.

Twoim zadaniem jest odpowiednie skonfigurowanie bufora strumienia «stdout» z użyciem procedury setvbuf(3) oraz zaimplementowanie metody zapisu z użyciem «writev».

Przy pomocy skryptu powłoki «writeperf.sh» porównaj wydajność wymienionych wcześniej metod zapisu. Uzasadnij przedstawione wyniki. Miej na uwadze liczbę wywołań systemowych (należy to zbadać posługując się narzędziem strace(1) z opcją «-c») oraz liczbę kopii danych wykonanych celem przesłania zawartości linii do buforów dysku.

Zadanie 7. Program «id» drukuje na standardowe wyjście **tożsamość**, z którą został utworzony, np.:

```
$ id
uid=1000(cahir) gid=1000(cahir) groups=1000(cahir),20(dialout),24(cdrom),25(floppy),
27(sudo),29(audio),30(dip),44(video),46(plugdev),108(netdev),123(vboxusers),999(docker)
```

Uzupełnij procedurę «getid» tak by zwracała identyfikator użytkownika getuid(2), identyfikator grupy getgid(2) oraz tablicę identyfikatorów i liczbę grup dodatkowych getgroups(2). Nie możesz z góry założyć liczby grup, do których należy użytkownik. Dlatego należy stopniowo zwiększać rozmiar tablicy «gids» przy pomocy realloc(3), aż pomieści rezultat wywołania «getgroups». Należy również uzupełnić ciało procedur «uidname» i «gidname» korzystając odpowiednio z getpwuid(3) i getgrgid(3).

Literatura

[1] „*Advanced Programming in the UNIX Environment*”
W. Richard Stevens, Stephen A. Rago
Addison-Wesley Professional; 3rd edition; 2013

[2] „*The Linux Programming Interface: A Linux and UNIX System Programming Handbook*”
Michael Kerrisk
No Starch Press; 1st edition; 2010

[3] „*Systemy operacyjne*”
Andrew S. Tanenbaum, Herbert Bos
Helion; wydanie czwarte; 2015

[4] „*Operating Systems: Three Easy Pieces*”
Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
https://pages.cs.wisc.edu/~remzi/OSTEP/

[2] https://en.wikipedia.org/wiki/Microbenchmark