Systemy operacyjne 2016

Lista zadań nr 3

Na zajęcia 20 października 2016

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Stallings (wydanie siódme): 3.1 – 3.5, 4.1, 4.6
- Tanenbaum (wydanie czwarte): 2.1, 2.2, 10.3

UWAGA! W trakcie prezentacji rozwiązań należy zdefiniować i wyjaśnić pojęcia, które zostały oznaczone **wytłuszczoną** czcionką.

Studenci są zachęcani do przeprowadzania dodatkowych eksperymentów związanych z treścią zadań i dzieleniem się obserwacjami z resztą grupy. Proszę najpierw korzystać z podręcznika systemowego (polecenia `man` i `apropos`), a w razie potrzeby sięgać do zasobów Internetu. Głównym podręcznikiem do zajęć praktycznych jest „*The Linux Programming Interface: A Linux and UNIX System Programming Handbook*". Należy zapoznać się z treścią §2 w celach poglądowych, a resztę książki czytać w razie potrzeby. Bardziej wnikliwe wyjaśnienia zagadnień można odnaleźć w książce „*Advanced Programming in the UNIX Environment*".

Zadania wymagające użycia rzutnika, oznaczenie **(P)**, należy starannie przygotować w domu – najlepiej w postaci pliku tekstowego z listą poleceń do wykonania i komentarzami. Każde zadanie należy mieć właściwie przygotowane do prezentacji przed zajęciami.

Zadanie 1 (P). Opisz **mapę pamięci** (ang. *memory layout*) procesu w systemie `Linux x86-64`. Jak zaimplementowano izolację między procesem użytkownika, a jądrem systemu? Czemu na każdej platformie pierwsze kilka megabajtów przestrzeni użytkownika jest niedostępne dla programu? Używając narzędzia `pmap(1)` zaprezentuj zawartość przestrzeni adresowej wybranego procesu, po czym zidentyfikuj w niej **obraz** (ang. *process image*). Nie zapomnij wyjaśnić znaczenia kolumn wydruku!

Zadanie 2 (P). Zapoznaj się ze strukturami **bloku kontrolnego procesu** (ang. *Process Control Block*) i **wątku** (ang. *Thread Control Block*) jądra `FreeBSD` – odpowiednio proc[1] oraz thread[2]. Zidentyfikuj położenie **uchwytów** do zasobów i **atrybutów** z tabeli 2.1 (§2.1.6) w `PCB` oraz `TCB`. Na podstawie obserwacji wymień zasoby współdzielone przez wątki należące do jednego procesu.

Zadanie 3. Przedstaw automat opisujący **stan procesu** w systemie `Linux` (rysunek 4.16 z §4.6). Jakie akcje lub zdarzenia wymuszają zmianę stanów? Należy właściwie rozróżnić zdarzenia **synchroniczne** od **asynchronicznych**. Wyjaśnij, które przejścia mogą być rezultatem działań podejmowanych przez: jądro systemu operacyjnego, kod sterowników, proces użytkownika albo administratora.

Zadanie 4 (P). W systemach uniksowych istnieje pojęcie **hierarchii procesów**. Przetestuj działanie polecenia `ps` z opcjami: `-A`, `-a`, `-u`user, `-C`cmd. Dla wybranego procesu wyświetl jego **identyfikator**, **grupę**, **rodzica** oraz **właściciela**. Kto jest rodzicem procesu `init`? Wskaż, które z wyświetlonych zadań są **wątkami jądra**. Jakie jest znaczenie poszczególnych znaków w kolumnie `STAT`? Wyświetl drzewiastą strukturę procesów poleceniem `pstree` – które ze wskazanych zadań są wątkami?

[1] `http://bxr.su/FreeBSD/sys/sys/proc.h\#483`
[2] `http://bxr.su/FreeBSD/sys/sys/proc.h\#204`

Zadanie 5. Jaką rolę pełnią **sygnały** w systemach uniksowych. W jakich sytuacjach jądro **dostarcza** sygnały procesowi? Kiedy proces **odbiera sygnały**? Co musi zrobić proces by **wysłać sygnał** albo **obsłużyć sygnał**? Których sygnałów nie można **zignorować**? Podaj przykład, w którym obsłużenie sygnału `SIGSEGV` lub `SIGILL` może być świadomym zabiegiem programisty.

Zadanie 6 (P). Zaprezentuj metody wysyłania sygnałów z użyciem poleceń `kill`, `pkill` i `xkill` na programie `xeyes`. Który sygnał jest wysyłany domyślnie? Przy pomocy kombinacji klawiszy `CTRL+Z` wyślij `xeyes` sygnał `SIGSTOP`, po czym wznów jego wykonanie. Przeprowadź inspekcję pliku `/proc/${PID}/status` i wyświetl **maskę sygnałów** zgłoszonych procesowi (ang. *pending signals*). Pokaż jak będzie się zmieniać, gdy będziemy wysyłać wstrzymanemu procesowi kolejne sygnały, tj. `SIGUSR1`, `SIGUSR2`, `SIGHUP`, `SIGINT`. Co opisują pozostałe pola pliku `status` dotyczące sygnałów? Który sygnał zostanie dostarczony jako pierwszy po wybudzeniu procesu?

Zadanie 7. Wyjaśnij różnice w tworzeniu procesów w systemie `Linux` i `WinNT`, po czym rozważ zalety i wady obu rozwiązań. W kontekście funkcji **fork(2)** wytłumacz na czym polega mechanizm **kopiowania przy zapisie** (ang. *copy-on-write*)? System `Linux` implementuje `fork` przy pomocy wywołania systemowego **clone(2)**. Jak zatem utworzyć proces albo wątek? Zastanów się czemu mogą służyć kombinacje pozostałych flag wywołania `clone`.

Zadanie 8 (P). Zapoznaj się z poleceniami `strace` i `ltrace`. Uruchom wybrany program w trybie śledzenia wywołań systemowych i wywołań bibliotecznych[3]. Podłącz się do wybranego procesu i obserwuj jego działanie. Jak śledzić aplikacje złożone z wielu procesów lub wątków? Jak zliczyć ilość wywołań systemowych, które wykonał proces w trakcie swego wykonania? Jak obserwować wyłącznie pewien podzbiór wywołań systemowych, np. `open`, `read` i `write`?

[3]Konfiguracja systemu może wymagać użycia polecenia `sudo` do uruchomienia programów śledzących.