Systemy operacyjne (zaawansowane)

Lista zadań nr 3

Na zajęcia 26 października 2017

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Tanenbaum (wydanie czwarte): 2.1, 10.3, 11.4
- Stallings (wydanie siódme): 3.1 – 3.4, 4.1, 4.6

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. Można nie otrzymać punktów za zadanie w przypadku zbędnego przeciągania czasu odpowiedzi ze względu na problemy techniczne. Najlepiej przygotować sobie skrypt, który z użyciem programu `xrandr`[1] ustawi rozdzielczość ekranu wbudowanego na 1024×768 i sklonuje go na zewnętrzne złącze VGA lub HDMI. Dla programu terminala należy wybrać dużą czcionkę (max. 40 linii w trybie pełnoekranowym) i kontrastowe kolory.

Zadanie 1. Przedstaw automat opisujący **stan procesu** w systemie `Linux` (rysunek 4.16 z §4.6). Jakie akcje lub zdarzenia wymuszają zmianę stanów? Uwzględnij również opuszczanie przez proces stanu **zombie**. 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 2. Wyjaśnij różnice w tworzeniu procesów w systemie `Linux` i `WinNT`. Rozważ zalety i wady obu rozwiązań. Czy tworzenie procesów poprzez klonowanie może być użyteczne z punktu widzenia projektanta oprogramowania? Czy ładowanie programów i tworzenie procesów powinno być osobną funkcją jądra? Naszkicuj przebieg akcji podejmowanych przez jądro przy obsłudze wywołań systemowych **fork(2)** i **exec(2)**. W jaki sposób jądro optymalizuje tych funkcji z użyciem mechanizmu **kopiowania przy zapisie** (ang. *copy-on-write*) i **stronicowania na żądanie** (ang. *demand paging*)?

Zadanie 3. Jaką rolę pełnią **sygnały** w systemach uniksowych? W jakich sytuacjach jądro wysyła sygnał procesowi? Kiedy jądro **dostarcza** sygnały do procesu? Co musi zrobić proces by **wysłać sygnał** albo **obsłużyć sygnał**? Których sygnałów nie można **zignorować** i dlaczego? Podaj przykład, w którym obsłużenie sygnału `SIGSEGV` lub `SIGILL` może być świadomym zabiegiem programisty.

[1] `https://wiki.archlinux.org/index.php/xrandr`

Zadanie 4 (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 5 (P). W systemach uniksowych istnieje pojęcie **hierarchii procesów**. Uruchom polecenie `ps -eo user,pid,pgid,tid,pri,stat,wchan,cmd`. Na wydruku zidentyfikuj **identyfikator**, **grupę**, **rodzica** oraz **właściciela** procesu. 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 zadań są wątkami?

Zadanie 6 (P). Do czego służy system plików proc[2] w systemie `Linux`? Zaprezentuj zawartość przestrzeni adresowej X-serwera[3] wyświetlając plik `/proc/`*$PID*[4]`/maps`, po czym zidentyfikuj w niej poszczególne *zasoby pamięciowe* tj. stos, stertę, **segmenty programu**, **pamięć anonimową**, **pliki odwzorowane w pamięci**, itp. Nie zapomnij wyjaśnić znaczenia kolumn wydruku!

Zadanie 7 (P). Uruchom aplikację `firefox` i przy pomocy programu `lsof` wyświetl *zasoby plikowe* należące do procesu przeglądarki. Podaj znaczenie poszczególnych kolumn wykazu i zidentyfikuj, które z wymienionych zasobów są **zwykłymi plikami**, **katalogami**, **urządzeniami**, **gniazdami** (sieciowymi lub domeny uniksowej), **potokami**. Przechwyć wyjście z programu `lsof` przed i po otwarciu wybranej strony w nowej zakładce, po czym wyświetl różnice poleceniem `diff -u`.

Zadanie 8 (P). Zapoznaj się z poleceniami `strace` i `ltrace`. Uruchom wybrany program w trybie śledzenia wywołań systemowych i wywołań bibliotecznych[5]. 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`?

[2] `http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html`
[3] `https://en.wikipedia.org/wiki/X_Window_System`
[4] Zastąp *$PID* identyfikatorem procesu `Xorg`!
[5] Konfiguracja systemu może wymagać użycia polecenia `sudo` do uruchomienia programów śledzących.