Systemy komputerowe

Lista zadań nr A

Na zajęcia 20–23 maja 2019

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Tanenbaum (wydanie czwarte): 2.1, 10.3, 11.4
- Stallings (wydanie dziewiąte): 3.1 – 3.4, 4.1, 4.4, 4.6

UWAGA! W trakcie prezentacji należy być gotowym do zdefiniowania pojęć oznaczonych **wytłuszczoną** czcionką.
Zachęcamy 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. W przypadku zbędnego przeciągania czasu odpowiedzi ze względu na problemy techniczne prowadzący ma prawo skreślić zadanie i postawić jeden punkt ujemny.
Wskazówka: Przygotuj 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 (około 32 wierszy w trybie pełnoekranowym), białe tło i czarny kolor pierwszoplanowy. Starannie przetestuj swoją konfigurację przed zajęciami!

Zadanie 1. Na podstawie rysunku 4.15 z §4.6 przedstaw **stany procesu** w systemie *Linux*. Jakie akcje lub zdarzenia **synchroniczne** i **asynchronicznych** wyzwalają zmianę stanów? Kiedy proces opuszcza stan **zombie**? 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* (§11.4.3). W jaki sposób klonowanie procesów może być użyteczne z punktu widzenia projektanta oprogramowania? Naszkicuj przebieg akcji podejmowanych przez jądro w trakcie obsługi funkcji **fork** i **exec**. Na czym polega optymalizacja implementacji klonowania procesów z użyciem mechanizmu **kopiowania przy zapisie** (ang. *copy-on-write*)?

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.

Zadanie 4 (P). Uruchom program «`xeyes`» po czym użyj na nim polecenia «`kill`», «`pkill`» i «`xkill`». Który sygnał jest wysyłany domyślnie? Przy pomocy kombinacji klawiszy «`CTRL+Z`» wyślij «`xeyes`» sygnał «`SIGSTOP`», a następnie wznów jego wykonanie. Przeprowadź inspekcję pliku «`/proc/$PID`[2]`/status`» i wyświetl **maskę sygnałów** zgłoszonych procesowi (ang. *pending signals*). Pokaż jak będzie się zmieniać, gdy będziemy wysyłać wstrzymanemu procesowi po kolei: «`SIGUSR1`», «`SIGUSR2`», «`SIGHUP`» i «`SIGINT`». Co opisują pozostałe pola pliku «`status`» dotyczące sygnałów? Który sygnał zostanie dostarczony jako pierwszy po wybudzeniu procesu?

[1] `https://wiki.archlinux.org/index.php/xrandr`
[2] Zastąp $PID identyfikatorem uruchomionego procesu!

Zadanie 5 (P). W systemach uniksowych wszystkie procesy są związane relacją **rodzic-dziecko**. 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 reprezentację **hierarchii procesów** poleceniem `pstree` – które z zadań są wątkami?

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

Zadanie 7 (P). Używając programu «`lsof`» wyświetl **zasoby plikopodobne** podpięte do procesu przeglądarki «`firefox`». Wyjaśnij znaczenie poszczególnych kolumn wykazu, po czym zidentyfikuj **pliki zwykłe**, **katalogi**, **urządzenia**, **gniazda** (sieciowe lub domeny uniksowej) i **potoki**. Przekieruj wyjście z programu «`lsof`», przed i po otwarciu wybranej strony, odpowiednio do plików «`before`» i «`after`». Czy poleceniem «`diff -u before after`» jesteś w stanie zidentyfikować nowo utworzone połączenia sieciowe?

Zadanie 8 (P). Uruchom[5] polecenie «`find /etc`» pod kontrolą programu `strace`[6]. Co pojawia się na wyjściu programu? Następnie podłącz się do istniejącego procesu (np. edytora tekstowego) i pokaż jak interakcja z programem wymusza komunikację z jądrem systemu. 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] `http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html`
[4] `https://en.wikipedia.org/wiki/X_Window_System`
[5] Konfiguracja systemu może wymagać użycia polecenia «`sudo`» do uruchomienia programów śledzących.
[6] `http://man7.org/linux/man-pages/man1/strace.1.html`