Systemy operacyjne

Lista zadań nr 1

Na zajęcia 15–16 października 2019

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Arpaci-Dusseau: Introduction[1]
- Tanenbaum (wydanie czwarte): 1.1, 1.2, 1.4, 1.5

UWAGA! W trakcie prezentacji należy być gotowym do zdefiniowania pojęć oznaczonych **wytłuszczoną** czcionką. Brak przygotowania będzie skutkować przyznaniem punktów motywujących do cięższej pracy.

UWAGA! Rozwiązanie zadania oznaczonego **(P)** należy pokazać z użyciem rzutnika posiadającego wejście `HDMI`. Przed zajęciami należy przygotować sobie skrypt wykorzystujący narzędzie «`xrandr`» do klonowania sygnału z wyświetlacza wbudowanego na zewnętrzny ekran. Terminal **musi** używać dużej czytelnej czcionki i kontrastowych kolorów.

Zadanie 1. Wyjaśnij różnice między **powłoką** (ang. *shell*), **system operacyjnym** i **jądrem systemu operacyjnego** (ang. *kernel*). W tym celu dobierz kilka przykładów powszechnie wykorzystywanego oprogramowania. Jakie są główne zadania systemu operacyjnego z punktu widzenia programisty?

Zadanie 2. Czym jest **zadanie** w **systemach wsadowych**? Jaką rolę pełni **monitor**? Na czym polega **planowanie zadań**? Zapoznaj się z rozdziałem „System Supervisor" dokumentu IBM 7090/7094 IBSYS Operating System[2]. Wyjaśnij pobieżnie znaczenie poleceń **języka kontroli zadań** (ang. *Job Control Language*) użytych na rysunku 3 na stronie 13. Do jakich zastosowań używa się dziś systemów wsadowych?

Wskazówka: Bardzo popularnym systemem realizującym szeregowanie zadań wsadowych jest SLURM[3].

Zadanie 3. Jaka była motywacja do wprowadzenia **wieloprogramowych** systemów wsadowych? W jaki sposób wieloprogramowe systemy wsadowe wyewoluowały w systemy z **podziałem czasu** (ang. *time-sharing*)? Podaj przykład historycznego systemu **interaktywnego**, który nie jest wieloprogramowy.

Zadanie 4. Wymień mechanizmy sprzętowe niezbędne do implementacji **wywłaszczania** (ang. *preemption*). Jak działa **algorytm rotacyjny** (ang. *round-robin*)? Jakie zadania pełni **planista** (ang. *scheduler*) i **dyspozytor** (ang. *dispatcher*)? Który z nich realizuje **politykę**, a który **mechanizm**?

Zadanie 5. Zapoznaj się z podrozdziałem „The Elements of Operating-System Style"[4] książki „The Art of Unix Programming". Czemu system operacyjny powinien (a) umożliwiać szybkie tworzenie procesów i łatwą komunikację międzyprocesową (b) przechowywać dane w plikach tekstowych, a nie binarnych (c) udostępniać szereg narzędzi programistycznych (d) oferować bogaty wybór programów działających w linii poleceń?

Ściągnij ze strony przedmiotu archiwum «`so19_lista_1.tar.gz`», następnie rozpakuj i skompiluj źródła poleceniem «`make`».

Zadanie 6 (P). Uruchom program «`1_ls`» pod kontrolą narzędzia «`ltrace -S`». Na podstawie śladu wykonania programu zidentyfikuj, które z **wywołań systemowych** są używane przez procedury: «`opendir`», «`readdir`», «`printf`» i «`closedir`». Do czego służy wywołanie systemowe «`brk`»? Używając debuggera «`gdb`» i polecenia «`catch syscall brk`» zidentyfikuj, która funkcja używa «`brk`».

Zadanie 7 (P). Pod kontrolą narzędzia «`strace`» uruchom program «`2_cat`» korzystający bezpośrednio z wywołań systemowych do interakcji ze **standardowym wejściem i wyjściem**. Pokaż, że program oczekuje na odczyt na **deskryptorze pliku** 0 i pisze do deskryptora 1. Naciśnij kombinację klawiszy «`CTRL+D`» kończąc wejściowy strumień danych – co zwróciło «`read`»? Zmodyfikuj program tak, by czytał z pliku podanego w linii poleceń. Co się stanie, jeśli przekażesz **ścieżkę** do katalogu zamiast do pliku regularnego?

[1] `http://pages.cs.wisc.edu/~remzi/OSTEP/intro.pdf`
[2] `http://bitsavers.org/pdf/ibm/7090/C28-6248-7_v13_IBSYS_Dec66.pdf`
[3] `https://slurm.schedmd.com/SC17/SlurmOverviewSC17.pdf`
[4] `http://www.catb.org/~esr/writings/taoup/html/ch03s01.html`