Systemy operacyjne 2016

Lista zadań nr 1

Na zajęcia 16 i 23 maja 2016

Pewna trudność w zajęciach z „*Systemów operacyjnych*" polega na opanowaniu sprawnego wyszukiwania treści w podręcznikach do wykładu, zasobach Internetu, w podręczniku systemowym (poleceniem `man`), plikach nagłówkowych w katalogu `/usr/include` (poleceniem `grep` lub `ack`) i źródłach jądra `FreeBSD`[1]. Posiłkowanie się hasłami z polskojęzycznej Wikipedii jest niewskazane ze względu na liczne błędy merytoryczne, które tam występują.

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Stallings (szóste wydanie): 2.2 – 2.5, 4.3
- Tanenbaum (czwarte wydanie): 1.2, 1.5, 1.7, 7 (fragmenty)

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

Zadanie 1. Podaj przykłady i wyjaśnij różnice między **powłoką**, **system operacyjnym** i **jądrem systemu operacyjnego**. Jakie są główne zadania systemu operacyjnego? Wymień podstawowe funkcjonalności jądra systemu operacyjnego.

Zadanie 2. Bardzo ważną zasadą przy projektowaniu oprogramowania, a w tym systemów operacyjnych, jest rozdzielenie **mechanizmu** od **polityki**. Wyjaśnij te pojęcia odnosząc się do powszechnie występujących rozwiązań, np. rozważ klasyczny klucz i kartę magnetyczną do otwierania drzwi.

Zadanie 3. W standardowej bibliotece języka `C` znajdują się funkcje `setjmp` i `longjmp` służące do przeprowadzania **nielokalnych skoków**. Wyjaśnij na przykładzie pseudokodu jak korzysta się z tego mechanizmu. Zauważ, że wyżej wymienione funkcje częściowo realizują **przełączanie kontekstu**. Jakie dane przechowywane są w strukturze `jmp_buf`? Skorzystaj z podręcznika systemowego i pliku nagłówkowego `setjmp.h`.

Zadanie 4. Dla każdego z podanych **wywołań systemowych** podaj kilka warunków, przy których mogą się one zakończyć niepowodzeniem: `fork`, `kill`, `mount`, `unlink`, `read`. W jaki sposób system operacyjny komunikuje programiście przyczynę niepowodzenia?

Zadanie 5. Czy jest jakakolwiek przyczyna, dla której użytkownik mógłby chcieć użyć niepustego katalogu jako **punktu montażowego**? Jeśli tak, to w jakich warunkach? Podaj przykłady i zastosowania **atrybutów punktów montażowych**.

Zadanie 6. Wyjaśnij główne różnice między **urządzeniami blokowymi** i **urządzeniami znakowymi**. Posiłkując się zawartością katalogu `/dev` podaj przykłady obydwu typów urządzeń. Które z **operacji na plikach** nie mają sensu dla plików urządzeń znakowych?

Zadanie 7. Czym jest **zadanie** w **systemach wsadowych**? Jaką rolę pełni **monitor**? Na czym polega **planowanie zadań**? Wymień najważniejsze polecenia wybranego **języka kontroli zadań** (ang. *Job Control Language*) i wyjaśnij do czego są służą. Czy w dzisiejszych czasach używa się systemów wsadowych. Jeśli tak, to do jakich zastosowań?

[1] `http://fxr.watson.org/fxr/source/`

Zadanie 8. Jaka była motywacja do wprowadzenia **wieloprogramowych** systemów wsadowych? W jaki sposób wieloprogramowe systemy wsadowe wyewoluowały w systemy z **podziałem czasu**? Czy **interaktywne** systemy operacyjne muszą być wieloprogramowe? Jeśli nie, to podaj przykład.

Zadanie 9. Podaj główne motywacje projektantów systemów operacyjnych do wprowadzenia **procesów** i **wątków**? Wymień główne różnice między nimi – rozważ współdzielone **zasoby**.

Zadanie 10. Wymień mechanizmy sprzętowe niezbędne do implementacji **wywłaszczania**. Wyjaśnij jak użyć **algorytmu rotacyjnego** (ang. *round-robin*) do implementacji wielozadaniowości z wywłaszczaniem. Za co odpowiada **planista** (ang. *scheduler*), a za co **dyspozytor** (ang. *dispatcher*)?

Zadanie 11. Na podstawie artykułu „Anatomy of the Linux kernel[2]" opowiedz z jakich głównych komponentów składa się **monolityczne jądro** Linuksa. Czym charakteryzuje się jądro zorganizowane w **warstwy**? Wymień poważne wady architektury monolitycznej często przytaczane w literaturze. Które z tych wad można zniwelować dzięki użyciu **modułów jądra** oraz interfejsów typu `FUSE`[3]?

Zadanie 12. Jądro systemu `NetBSD` jest łatwo przenośne (ang. *portable*) dzięki dobrej implementacji **warstwy abstrakcji sprzętu** (ang. *hardware abstraction layer*). Do czego służy `HAL` i jakie funkcjonalności pokrywa? Jaka jest zależność między `HAL`, a **sterownikami urządzeń**?

Zadanie 13. Posiłkując się listą[4] wybierz po jednym jądrze systemu operacyjnego:

- **czasu rzeczywistego**,
- dla **sieci sensorów**,
- dla **systemów wbudowanych**.

Wyjaśnij w jakich warunkach korzysta się z tych systemów i w jaki sposób różnią się one od standardowych jąder systemów operacyjnych przeznaczonych dla komputerów osobistych.

Zadanie 14. Wyjaśnij czym jest architektura **klient-serwer** w kontekście systemów operacyjnych. Jakie są główne różnice między **mikrojądrem**, a jądrem monolitycznym? Wymień wady i zalety tej architektury. Na podstawie publikacji „On μ-Kernel Construction[5]" podaj obszary funkcjonalności, które musi implementować każde mikrojądro.

Zadanie 15. Podaj przykłady **usług** i funkcjonalności jądra monolitycznego, które mogą być zrealizowane jako procesy poziomu użytkownika w systemie operacyjnym z mikrojądrem. `Windows NT` został pierwotnie zaprojektowany jako mikrojądro. W trakcie rozwoju zdecydowano o migracji do architektury **jądra hybrydowego**. Dlaczego tak się stało? Jakie korzyści to przyniosło?

Zadanie 16. Czemu służy **wirtualizacja**? Wymień dwa typy **monitorów maszyn wirtualnych**[6] i na podstawie diagramu wyjaśnij różnice między nimi. Wyjaśnij pojęcia **systemu gościa** (ang. *guest OS*) oraz **systemu gospodarza** (ang. *host OS*). Czym się różni **emulacja** od **symulacji** systemu?

[2] `https://www.ibm.com/developerworks/library/l-linux-kernel/`
[3] `http://lxr.free-electrons.com/source/Documentation/filesystems/fuse.txt`
[4] `https://en.wikipedia.org/wiki/List_of_operating_systems`
[5] `http://os.itec.kit.edu/downloads/publ_1995_liedtke_ukernel-construction.pdf`
[6] czasami zwanych hiper-zarządcami (ang. *hypervisor*)