Systemy operacyjne

Ćwiczenia 2

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Stallings: 2.2 – 2.5, 4.3 (6-ta edycja)
- Tanenbaum: 1.2, 1.5, 1.7, 7 (fragmenty)
- Silberschatz: 2.6, 2.7

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

Zadanie 1

Wyjaśnij koncepcję **systemów wsadowych.** Czym jest **zadanie**? Podaj rolę **monitora**. Czym jest **planowanie zadań**? Jakich mechanizmów sprzętowych wymagają takie systemy? Posługując się informacjami nt. **języka kontroli zadań** (ang. *Job Control Language*) – opisz jego najważniejsze polecenia i wyjaśnij do czego są potrzebne. Czy w dzisiejszych czasach używa się systemów wsadowych, a jeśli tak to do czego?

Zadanie 2

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

Zadanie 3

Przedstaw krótko historię kształtowania się **procesów** i **wątków** w systemach operacyjnych. Zauważ, że rozwój ten przebiegał zarówno na płaszczyźnie programowej jak i sprzętowej. Co było główną motywacją projektantów systemów operacyjnych do wprowadzenia procesów i wątków? Czym jest **przestrzeń adresowa**? Jakie są główne różnice między procesami, a wątkami?

Zadanie 4

Zdefiniuj pojęcie **wywłaszczania**. Podaj mechanizmy sprzętowe niezbędne do implementacji wywłaszczania. Wyjaśnij jak **algorytm rotacyjny** (ang. *round-robin*) jest używany do implementacji **wielozadaniowości** z wywłaszczaniem. Za co odpowiada **planista** (ang. scheduler), a za co **dyspozytor** (ang. dispatcher)?

Zadanie 5

Z jakich głównych **komponentów** składa się **monolityczne jądro** Linuksa – znajdź uproszczony diagram i zaprezentuj go. Wymień kilka interfejsów (ang. *Application Programming Interface*) zaimplementowanych na potrzeby jądra (a nie programów użytkownika). Co to znaczy, że jądro jest zorganizowane w **warstwy**? W literaturze często przytacza się poważne wady architektury monolitycznej – jakie? Jak te problemy rozwiązują **moduły jądra** oraz interfejsy typu **FUSE**?

Zadanie 6

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 7

Wyjaśnij czym jest **architektura klient-serwer** w kontekście systemów operacyjnych. Jakie są główne różnice między **mikrojądrem**, a **jądrem monolitycznym**? Wymień wady i zalety mikrojądra w porównaniu z jądrem monolitycznym. Podaj obszary funkcjonalności, które musi implementować każde mikrojądro[1].

Zadanie 8

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 9

Wymień trzy klasy **maszyn wirtualnych** i wyjaśnij różnice między nimi na podstawie diagramu architektury. Do której klasy należy *VMWare*, *VirtualBox*, *Xen*, *QEMU*, *JavaVM*, Hyper-V (mogą do więcej niż jednej) ? Wyjaśnij pojęcia **guest OS**, **host OS**, **hypervisor**. Dla każdej klasy podaj przykłady zastosowań. Czym się różni **emulacja** od **symulacji** systemu?

[1] Według publikacji: On µ-Kernel Construction, Jochen Liedtke.