# Systemy operacyjne 2017

Lista zadań nr 1

Na zajęcia 12 października 2017

Pewna trudność w zajęciach z „*Systemów operacyjnych*" polega na opanowaniu sprawnego wyszukiwania treści w podręcznikach do wykładu, zasobach Internetu, w podręczniku systemowym (poleceniem `man`[1]), plikach nagłówkowych w katalogu `/usr/include`[2] (poleceniem `grep` lub `ack`) i źródłach jądra `FreeBSD`[3]. 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:

- Tanenbaum (wydanie czwarte): 1.1, 1.2, 1.4, 1.5
- Stallings (wydanie siódme): 2.1 – 2.4, 13.3, 13.4

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

**Zadanie 1.** Na podstawie wybranych przez siebie przykładów wyjaśnij różnice między **powłoką** (ang. *shell*), **system operacyjnym** i **jądrem systemu operacyjnego** (ang. *kernel*). Jakie są główne zadania systemu operacyjnego z punktu widzenia programisty?

**Zadanie 2.** Na podstawie dokumentacji[4] wymień składowe **pakietu** `deb` ze szczególnym uwzględnieniem zawartości pliku `control`. Porównaj zarządzanie zainstalowanym oprogramowania z użyciem pakietów i instalatorów znanych z systemów nieuniksowych. Weź pod uwagę proces pobierania, weryfikacji, instalacji, konfiguracji i odinstalowania oprogramowania.

**Zadanie 3.** Bardzo ważną zasadą przy projektowaniu oprogramowania, w tym systemów operacyjnych, jest rozdzielenie **mechanizmu** od **polityki**. Wyjaśnij te pojęcia odnosząc się do powszechnie występujących rozwiązań, np. otwieranie drzwi klasycznym kluczem versus kartą magnetyczną.

**Zadanie 4.** 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?

**Zadanie 5.** 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 systemu **interaktywnego**, który nie jest wieloprogramowy.

**Zadanie 6.** Podaj główne motywacje projektantów systemów operacyjnych do wprowadzenia **procesów** i **wątków**? Wymień główne różnice między nimi – rozważ współdzielone **zasoby**.

[1] `https://www.freebsd.org/cgi/man.cgi`
[2] `http://bxr.su/FreeBSD/include/`
[3] `http://bxr.su/FreeBSD/sys/`
[4] `http://tldp.org/HOWTO/html_single/Debian-Binary-Package-Building-HOWTO/`

**Zadanie 7.** Wymień mechanizmy sprzętowe wymagane do implementacji **wywłaszczania** (ang. *preemption*). Jak użyć **algorytmu rotacyjnego** (ang. *round-robin*) do implementacji wielozadaniowości z wywłaszczaniem? Jakie zadania pełni **planista** (ang. *scheduler*) i **dyspozytor** (ang. *dispatcher*)? Który z nich realizuje politykę, a który mechanizm?

**Zadanie 8.** Istnieją systemy operacyjne, które znacznie różnią się od tych znanych z komputerów osobistych (ang. *Personal Computer*). Scharakteryzuj krótko poniższe jądra:

- **czasu rzeczywistego** (ang. *real-time OS*) – `FreeRTOS`[5],
- dla **sieci sensorów** (ang. *OS for wireless sensor networks*) – `TinyOS`[6] (§13.3),
- dla **systemów wbudowanych** (ang. *embedded system OS*) – `eCos`[7] (§13.4).

Jak wspierane platformy sprzętowe i zakres zastosowań wpłynęły na zbiór funkcjonalności udostępnianych przez powyższe jądra? Jakie korzyści przynosi **konfigurowalna** architektura systemu `eCos`?

[5] `http://www.freertos.org`
[6] `http://www.tinyos.net`
[7] `http://ecos.sourceware.org`