Systemy operacyjne

Wojciech Kwedlo
Wydział Informatyki PB, p. 205
w.kwedlo@pb.edu.pl
aragorn.pb.bialystok.pl/~wkwedlo/Dydaktyka.html

Konsultacje: *Wtorek, Środa: 12:15-13:45, pokój 205*

Przebieg zajęć i sposób ich zaliczenia

- Wykład: Podstawowe informacje [w zasadzie teoretyczne :-( ] na temat budowy i funkcjonowanie systemów operacyjnych.
- Slajdy z wykładu: W całości dostępne na mojej stronie www.
  - Ale uwaga, proszę o ściągnięcie ze strony, a nie od Państwa kolegów (i koleżanek) z poprzednich lat, w treści wykładu będą zmiany.
- Zaliczenie wykładu: Egzamin pisemny, 20-30 pytań testowych, nie przewiduję zadań programistycznych.

- Pracownia specjalistyczna:
  - Pierwsze 50%. Programowanie z wykorzystaniem wywołań systemowych Linuksa (pliki, procesy, demony, sygnały)
  - Drugie 50%. Programowanie współbieżne z wykorzystaniem wątków (POSIX threads)
  - W planie cztery wejściówki i dwa projekty.

Literatura - ogólne pozycje na temat systemów operacyjnych

- Wykład (i pytania na egzamin) opracowane są na podstawie trzech znanych w Polsce i na świecie podręczników
- A. Silberschatz, P. B. Galvin, Podstawy systemów operacyjnych, wydanie 7, WNT, Warszawa, 2003.
- A.S. Tanenbaum, Systemy Operacyjne, Wydanie III, Wydawnictwo Helion, 2010. ~100 zł.
- W. Stallings, Systemy operacyjne. Struktura i zasady budowy. PWN, 2006. ~40 zł
- ***Gorąco polecam powyższe pozycje.***

Literatura - programowanie współbieżne (także pracownia specjalistyczna)

- Z. Weiss, T. Gruźlewski, Programowanie współbieżne i rozproszone w przykładach i zadaniach, WNT, Warszawa, 1993.
- M. Ben-Ari, Podstawy programowania współbieżnego i rozproszonego, WNT, Warszawa, 2009.
- Programowanie w Linuxie przy pomocy biblioteki wątków POSIX threads jest omówione w:
  - M. Mitchell, J. Oldham, A. Samuel, Linux - programowanie dla zaawansowanych, Wydawnictwo RM, 2002
  - J. Shapley Gray, Arkana: Komunikacja między procesami w Unixie, rozdz. 11.
  - R. Love, Linux. Programowanie systemowe, wydanie II, Helion 2014 – ***gorąco polecam***.

Literatura - opisy jąder różnych wersji Unixa

- Bach M., Budowa systemu operacyjnego UNIX, WNT Warszawa, 1995, Opis jądra Systemu V Release 4.
- U. Vahalia, Jądro systemu Unix - nowe horyzonty, WNT Warszawa,2001, Opis historii Unixa i różnych cech jądra systemów Unixowych (System V, BSD, Solaris, Mach).
- M. Beck, H. Bohm, M. Dziadzka, U. Kunitz, R. Magnus, D. Verworner, Linux Kernel - Jądro systemu, wydanie II, Wydawnictwo MIKOM, Warszawa, 2000, jądro Linuxa 2.0.x
- D. P. Bovet, M. Cesati, Linux kernel, Wydawnictwo RM, Warszawa, 2001, jądro Linuxa 2.2.x
- R. Love, Jądro Linuksa. Przewodnik, Wydanie 3, Helion, Warszawa 2014

Literatura - API wywołań systemowych Linuksa (wyłącznie pracownia specjalistyczna)

- R. Love, Linux. Programowanie systemowe, wydanie II, Helion 2014 – ***gorąco polecam***.
- M. Mitchell, J. Oldham, A. Samuel, Linux - programowanie dla zaawansowanych, Wydawnictwo RM, 2002
- Kurt Wall, Linux : programowanie w przykładach, Mikom, 2000.
- Michael K. Johnson, Erik W. Troan, Oprogramowanie użytkowe w systemie LINUX, WNT, 2000.

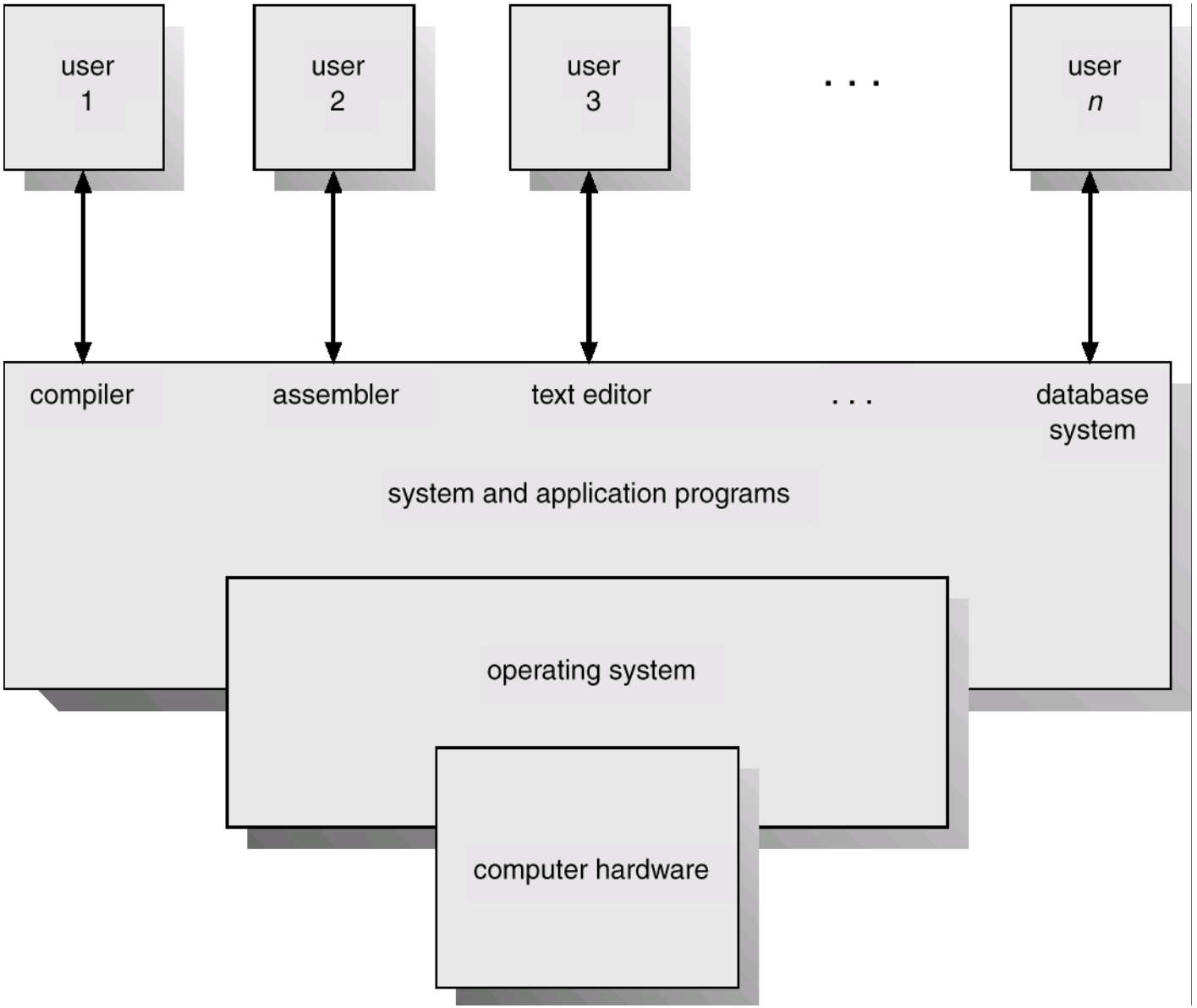
Wstępny plan wykładu

• Wstęp	2g.
• Struktury systemów komputerowych	2g.
• Procesy i Wątki	2g.
• Komunikacja międzyprocesowa	4g.
• Szeregowanie procesów (i wątków)	2g.
• Zarządzanie pamięcią i pamięć wirtualna	4g.
• Urządzenia pamięci masowej	2g.
• Systemy plików	4g.
• Bezpieczeństwo oraz ochrona	2g.
• Systemy wieloprocesorowe	2g.
• Systemy rozproszone	2g.

Czym jest system operacyjny

- Program kontrolujący wykonywanie aplikacji (programów użytkownika).
  - Umożliwia wykonywanie programów użytkownika. (być może wielu równocześnie). Jednocześnie kontroluje pracę tych programów, uniemożliwiając nieprawidłowe wykorzystanie komputera.
  - Dostarcza usługi umożliwiające dostęp do wejścia-wyjścia (pliki, ekran, klawiatura, sieć)
  - Zarządza zasobami maszyny (procesor, pamięć operacyjna, pamięć dyskowa).
  - Tworzy abstrakcje ułatwiające korzystanie z komputera, takie jak: plik, pamięć wirtualna, proces
- Cele stojące przed projektantem systemu
  - Przenośność: programy użytkowe działające na różnym sprzęcie.
  - Wydajność: efektywne wykorzystanie sprzętu (procesora,pamięci,we-wy).
  - Zapewnienie ochrony systemu oraz procesów (przed innymi procesami, niepowołanym dostępem, wirusami, ...)
  - Wygoda użytkownika.
  - Możliwość ewolucji systemu (poprawki etc.)
- Definicja: nie ma powszechnej definicji (“Wszystko, to co dostarcza producent, gdy kupuje się system operacyjny”)

System operacyjny na tle innych składników komputera



Jądro systemu operacyjnego (ang. kernel)

- Określenie co jest częścią systemu operacyjnego, a co nie, nie jest proste.
  - czy kompilator C jest częścią systemu ? edytor tekstu (np. notepad) ? Interpreter poleceń (ang. shell) w Unixie ?
- Dla potrzeb tego wykładu przez "system operacyjny" będziemy rozumieli jądro systemu, które możemy (z grubsza) scharakteryzować jako:
  - Część systemu przebywająca na stałe w pamięci.
  - Wykonuje się w trybie uprzywilejowanym.
  - Zarządza zasobami maszyny
  - Udostępnia interfejs programom użytkownika w postaci wywołań systemowych.
- Unix, Linux – jądro monolityczne: jeden olbrzymi program
  - Możliwość dynamicznego ładowania i usuwania fragmentów jądra, zwanych modułami
  - Interpreter poleceń i inne narzędzia są zwykłymi programami użytkownika (nazywane także programami systemowymi).
- Powyższy podział jest charakterystyczny dla systemów Unixowych, w świecie Microsoftu są odstępstwa (np. wiele funkcji z API Win32 wykonuje się na poziomie procesu użytkownika)

Perspektywa historyczna

- Na ewolucję oraz rozwój systemów operacyjnych olbrzymi wpływ miał czynnik ekonomiczny – ***relatywny (względny) koszt sprzętu oraz pracy ludzkiej***.
- Początkowo koszt komputera był fantastycznie wysoki w porównaniu z kosztem pracy ludzi z niego korzystających.
  - Główny cel stojący przed systemem, to maksymalizacja wykorzystania sprzętu (procesora, pamięci masowych drukarek).
  - Rozwiązania typu: przetwarzanie wsadowe, wieloprogramowość, spooling
- Obecnie koszt komputera jest bardzo niski w z kosztem pracy ludzi z niego korzystających.
  - Cel stojący przed systemem to stworzenie środowiska zapewniającego jak największą wygodę i produktywność użytkowników. Może to oznaczać rezygnację z kryterium maksymalnego wykorzystania sprzętu na rzecz kryterium minimalnego czasu odpowiedzi.
  - Praca interakcyjna (początkowo wielu użytkowników na jednej maszynie poprzez podział czasu procesora), praca interakcyjna z wieloma programami, graficzne interfejsy użytkownika + wielowątkowość

Generacje komputerów i systemów operacyjnych

- (1945-55) - Lampy próżniowe i brak systemu operacyjnego.
- (1955-65) – Tranzystory i proste systemy wsadowe
- (1965-80) – Układy scalone i wieloprogramowe systemy wsadowe (oraz systemy z podziałem czasu)
- 1980 – obecnie komputery PC.

Brak systemu operacyjnego

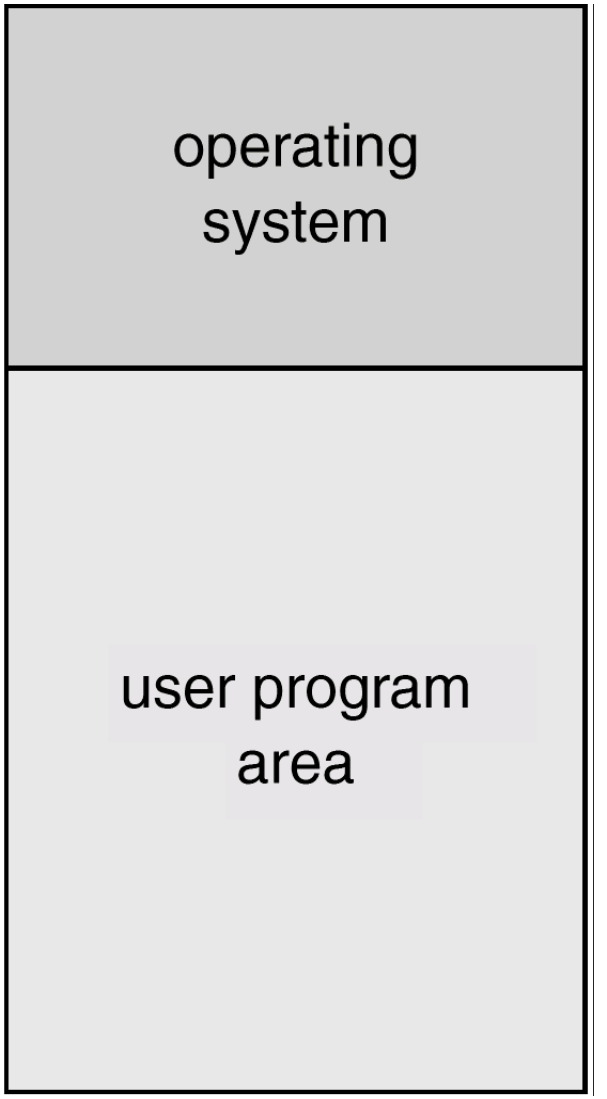
- Lata 40 – połowa lat 50, komputery zbudowane z lamp próżniowych
- Programista rezerwuje w specjalnym zeszycie czas komputera
- W zarezerwowanym czasie ma pełną kontrolę nad maszyną poprzez konsolę
  - Wczytuje program z czytnika kart perforowanych lub taśmy magnetycznej
  - Kompiluje program (w tym celu wczytuje kompilator np. Fortranu)
  - Uruchamia program..
  - Drukarka drukuje wyniki.
- Problemy:
  - Duża część czasu spędzana na przygotowaniu programu do uruchomienia
  - Nieefektywne wykorzystanie czasu pracy systemu.
- Zaleta: w przypisanym czasie mam pełną kontrolę nad komputerem

Proste systemy wsadowe

- Programista zostawia zadanie (ang. job) w postaci pliku kart perforowanych (program źródłowy, dane) operatorowi.
- Czynności o podobnym charakterze (np. Kompilacja) grupowane są razem
- Zadania łączone we wsad (ang. Batch) i wykonywane.
- Programista po jakimś czasie (kilka godzin) otrzymuje wyniki.
- Program monitora (protoplasta systemu operacyjnego)
  - Wczytuje program użytkownika do pamięci
  - Przekazuje mu sterowanie
  - Po zakończeniu pracy program użytkownika powraca do monitora
  - Może zawierać procedury obsługi wejścia wyjścia
- Zaleta: lepsze wykorzystanie (bardzo drogiego) sprzętu.
- Wada: programista czy też użytkownik nie ma kontroli nad wykonywaniem zadania (ale jego zdanie się nie liczy – względy ekonomiczne).

Prosty system wsadowy - jednoprogramowość

- Musimy przeznaczyć część pamięci na monitor
- Jeden program w pamięci.
- Pożądane wprowadzenie sprzętowej ochrony monitora przed programami użytkownika.
- W niektórych zastosowaniach biznesowych (banki ubezpieczenia) wejście-wyjście może zajmować dużą część czasu wykonania zadania.
- W takim przypadku procesor nie jest efektywnie wykorzystany.
- Potrzebne są mechanizmy pozwalające na jednoczesne wykorzystanie procesora i we-wy.

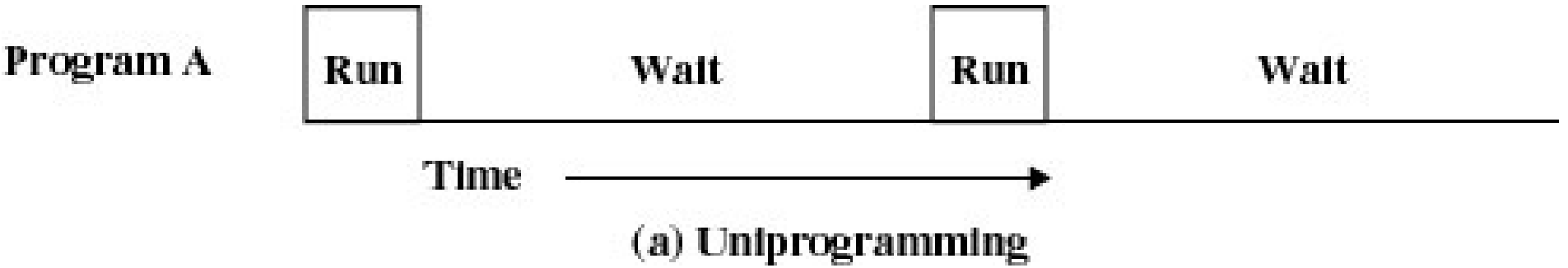


Spooling

- Simulateneous Peripheral Operation On-Line
- Wykorzystuje szybką pamięć dyskową.
- Dane i program z czytnika kart są przesyłane na dysk
- Program wczytany z dysku, wykonywany, wyniki są zapisywane na dysk
- Wyniki z dysku przesyłane są na drukarkę

- Wykonując zadanie i jednocześnie:
  - Wczytuj kod następnych zadań z czytnika kart na dysk
  - Drukuj wyniki już zakończonych zadań.
- Pula gotowych zadań na dysku – algorytmy szeregowania (ang. scheduling)

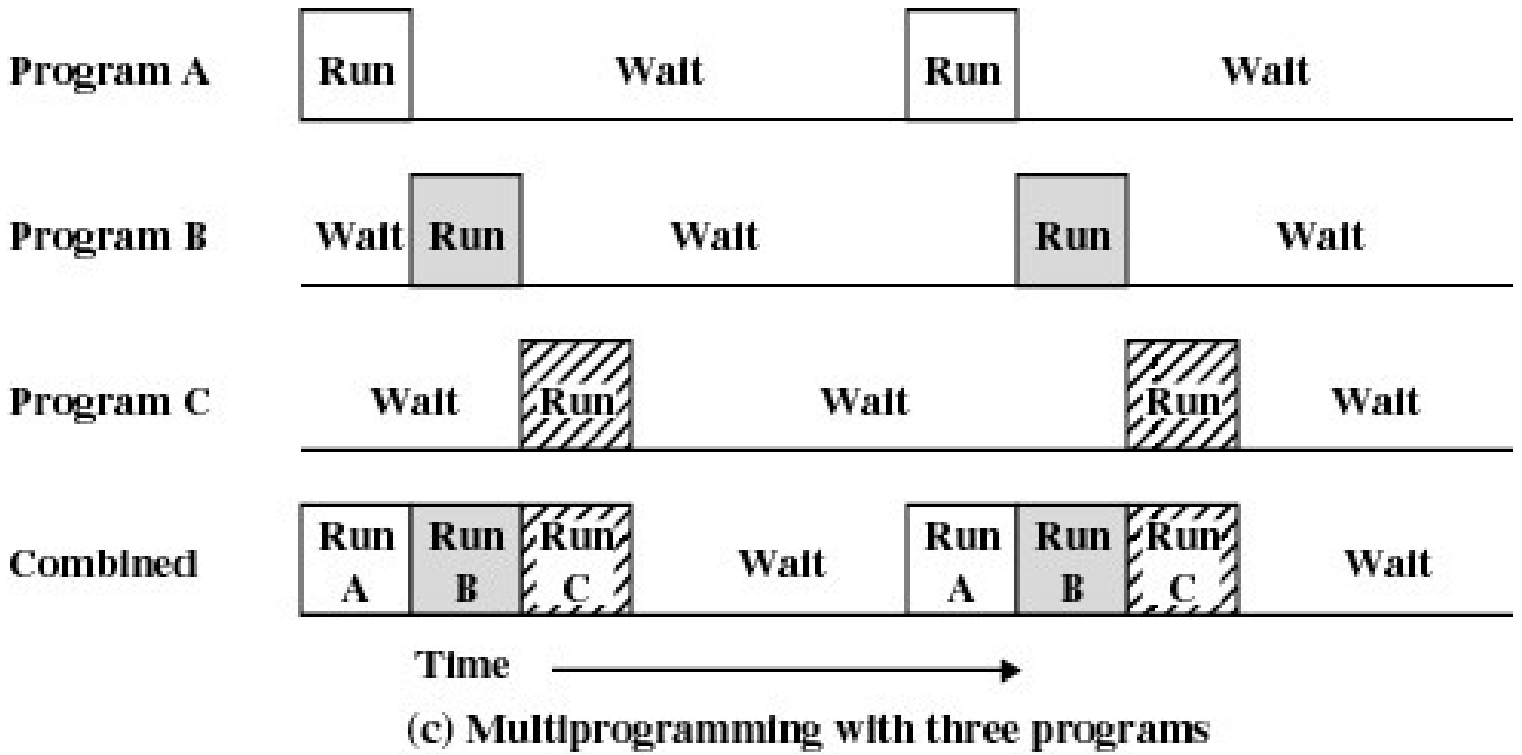
Wada jednoprogramowości



- Podczas wykonywania we-wy procesor jest bezczynny

- Rozwiązanie: wieloprogramowość (ang. Multiprogramming)
  - W pamięci przechowywanych jest kilka programów

Wieloprogramowość

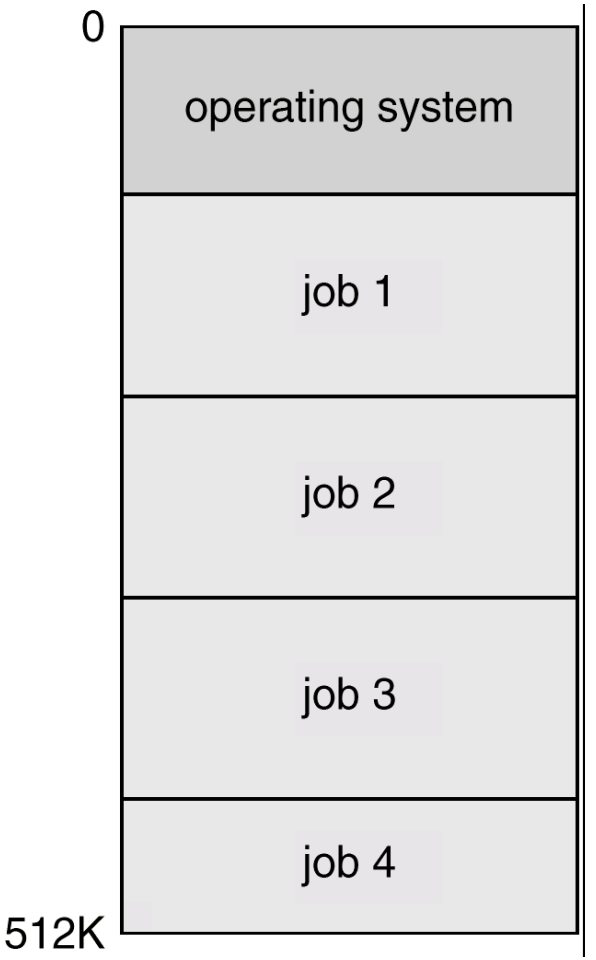


(c) Multiprogramming with three programs

- Kiedy zadanie zgłasza zapotrzebowanie na usługę systemu wymagającą oczekiwania, wykonywany jest inne zadanie
- Potrzebny jest specjalny hardware (przerwania, DMA, ochrona pamięci)

Wieloprogramowy system wsadowy

- Zapewnianie usług we-wy.
- Przydzielanie urządzeń we-wy zadaniom.
- Zadania wstępnie na dysku - zarządzanie pamięcią dyskową.
- Wybór zadania wczytywanego do pamięci (szeregowanie zadań)
- Przydzielenie procesora jednemu z zadań rezydującemu w pamięci
- Zarządzanie dostępną pamięcią



Współdzielenie procesora (ang. time sharing)

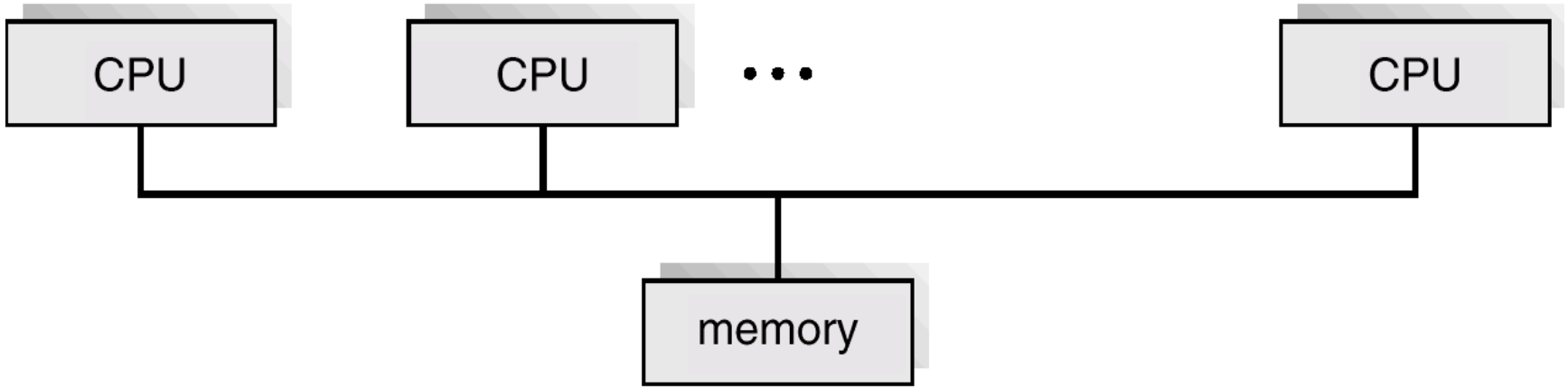
- W systemach wsadowych celem jest maksymalizacja przepustowości. Programista (użytkownik) nie ma kontroli nad wykonywaniem zadań.
  - Problem w przypadku np. błędu kompilacji.
  - Niektóre typy zadań wymagają pracy interakcyjnej (np. rezerwacja biletów)
  - A w między czasie relatywne koszty pracy ludzkiej względem kosztów sprzętu wzrosły, ale nie na tyle aby każdemu przydzielić dedykowany komputer.
- Podział czasu procesora
  - Wiele użytkowników korzystających z terminali (praca interakcyjna)
  - System otrzymuje polecenia kontrolne z terminala.
  - Procesor wykonuje na przemian programy poszczególnych użytkowników (redukcja mocy obliczeniowej dostępnej pojedynczemu użytkownikowi)

- Załadowany do pamięci i wykonywany w niej program nazywany jest ***procesem.***
- Nowe funkcje systemu operacyjnego: wymiana (ang. swapping), pamięć wirtualna.
- W systemach z podziałem czasu celem jest minimalizacja czasu odpowiedzi.

Systemy operacyjne dla komputerów osobistych

- Komputer dedykowany jednemu użytkownikowi.
- Mała cena sprzętu sprawia, że maksymalizacja wykorzystania procesora przestaje być głównym celem.
- Priorytetem jest wygoda użytkownika i minimalizacja czasu odpowiedzi.
- Systemy graficznego interfejsu użytkownika.
- Z przyczyn ekonomicznych pierwsze wersje sprzętu były prymitywnie proste. Nastąpiło stopniowe zaadoptowanie technologii rozwiniętych dla większych maszyn.
  - MS-DOS (brak podziału czasu, brak ochrony systemu)
  - Windows 3.1 (współdzielenie procesora wymagające współpracy procesów, pamięć wirtualna, brak ochrony)
  - Windows NT (współdzielenie procesora, pamięć wirtualna, pełna ochrona, mechanizmy bezpieczeństwa)

Systemy równoległe



- System ze współdzieloną pamięcią (ang. shared memory, tightly coupled system).
- Każdy procesor ma identyczny widok pamięci operacyjnej → kod (programu użytkownika, systemu) może się wykonywać na dowolnym procesorze.
- Architektura postaci Symmetric Multiprocessing (SMP) – wszystkie procesory są sobie równoważne (brak procesora wyróżnionego).
- Pozwala na wykonywanie wielu procesów jednocześnie
- Wspierana przez większość nowoczesnych systemów operacyjnych (Linux, Windows, Android, iOS, OS-X)
- Upowszechnione dzięki procesorom wielordzeniowym (ang. multicore)

Systemy czasu rzeczywistego (ang. real-time)

- Reakcja na zdarzenie musi się zakończyć przed upływem określonego czasu. (ang. *hard real-time*).
  - Innymi słowy: Poprawność systemu nie zależy tylko od poprawności uzyskanej odpowiedzi, ale także od czasu oczekiwania na tę odpowiedzi.
- Przykłady: odtwarzacz DVD, symulator samolotu, sterownik wtrysku paliwa, system naprowadzania rakiety.
- Z reguły system w ROM-ie, brak pamięci dyskowej.
- Konflikt z zasadą podziału czasu.
- Warunek hard real-time nie jest spełniony przez większość współczesnych systemów ogólnego przeznaczenia.
- Systemy klasy *soft real-time*: zadanie do obsługi w czasie rzeczywistym otrzymuje pierszeństwo nad pozostałymi zadaniami.
- Nie gwarantują nieprzekraczalnego czasu reakcji.
- Większość systemów Unixowych i Windowsowych spełnia te wymagania.

Systemy rozproszone (ang. distributed)

- Luźno związany (ang. loosely coupled) system komputerowy. Każdy procesor dysponuje własną pamięcią, niedostępną innym procesorom. Koordynacja i synchronizacja poprzez wymianę komunikatów

- Zalety
  - Współdzielenie zasobów (ang. resource sharing)
  - Przyspieszenie obliczeń.
  - Większa wiarygodność systemu.
  - Udostępnienie możliwości komunikacji.

Systemy rozproszone

- Sieciowe systemy operacyjne
  - Umożliwia współdzielenie plików, drukarek
  - Dostarcza mechanizm komunikacji
  - System na jednej maszynie wykonuje się niezależnie od pozostałych maszyn w sieci

- Rozproszone systemy operacyjne
  - Mniejsza autonomia poszczególnych maszyn
  - Sprawia wrażenie, że pojedynczy system operacyjny kontroluje pracę sieci (jednej wielkiej maszyny)
  - Obecnie przedmiot badań naukowych (Amoeba, Mach, Locus, systemy typu Grid ...)

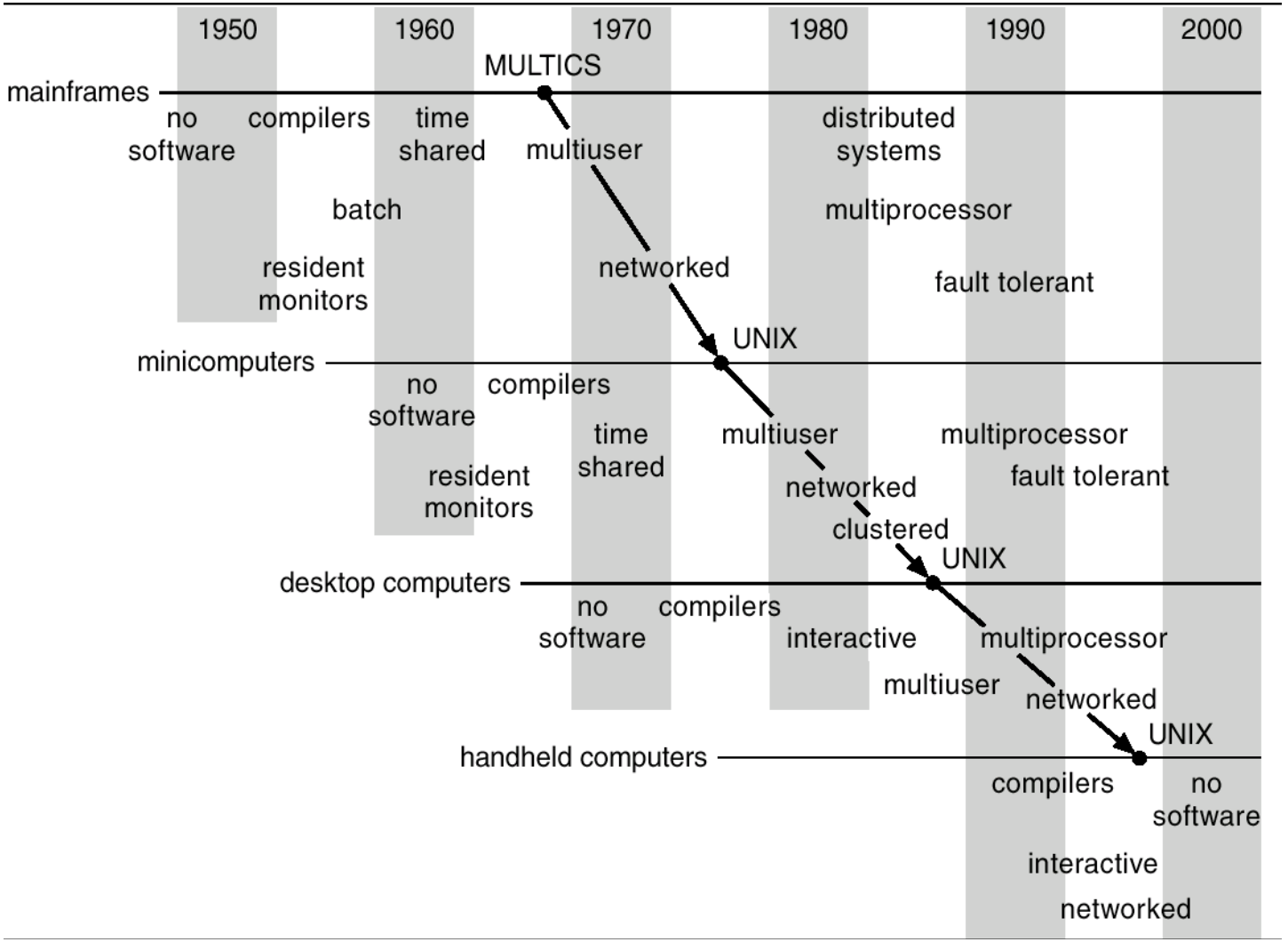
Systemy typu hand-held

- Tablety
- Telefony komórkowe
- Cechy charakterystyczne
  - Ograniczona pamięć
  - Wolne procesory (nacisk na niski pobór mocy)
  - Małe rozmiary wyświetlaczy, ekrany dotykowe (trochę inny interfejs – brak myszy)
  - Nowe cechy sprzętu (GPS, żyroskopy, ..) → nowe usługi systemu i nowe typy aplikacji (np. rzeczywistość rozszerzona)
- Przykłady: Apple iOS, Google Android.

Wbudowane systemy operacyjne

- Systemy dla urządzeń nie uznawanych za komputery i nie posiadających możliwości instalowania aplikacji przez użytkowników.
- Kuchenki mikrofalowe, odtwarzacze MP3, samochody, odbiorniki telewizyjne
- Mamy gwarancje że nigdy nie będzie działało niezaufane oprogramowanie
- Przykłady: QNX, VxWorks

Migracja cech i własności systemów operacyjnych



Systemy Operacyjne o otwartych źródłach

- „Open-source software (OSS) is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose.” (Wikipedia, Andrew M. St Laurent, Understanding Open Source and Free Software Licensing, 2008)
- Przeciwieństwo oprogramowania o zamkniętych źródłach (Windows).
  - Różnica z Microsoft Shared Source ?
- Najbardziej popularne GNU/Linux, FreeBSD
- Apple OS-X zawiera zarówno komponenty o zamkniętych jak i otwartych źródłach (jądro Darwin).
- Zalety
  - Społeczność zainteresowanych (zazwyczaj bezpłatnych) programistów
  - Większe bezpieczeństwo - „wiele oczu” śledzi źródła kodu
  - Wcale nie wyklucza zastosowań komercyjnych (RedHat, Canonical).