Systemy operacyjne

Wykład 1: Wprowadzenie

Cele wykładu

Widok na system operacyjny z punktu widzenia programisty.

Chcemy Was nauczyć:

- zestawu abstrakcji udostępnianych przez jądro:
  pliki, procesy, gniazda, odwzorowania w pamięć
- analizy interakcji programu z jądrem SO
- które elementy należą do SO, a które do jądra SO
- pobieżnie najważniejszych algorytmów jądra

Szczegóły implementacji algorytmów i struktur danych jądra zostawiamy na przedmiot “Struktura jądra UNIX”.

Dlaczego warto studiować systemy operacyjne?

- Pomost między sprzętem a oprogramowaniem
- Mają duży wpływ na wydajność całego systemu
- Podwaliny bezpieczeństwa komputerowego
- Nietrywialne algorytmy, struktury danych, problemy; np.:
  - pamięć wirtualna → wersjonowanie przestrzeni adresowych;
  - stos sieciowy → filtrowanie pakietów, routing;
  - systemy plików → odporność na uszkodzenia, szyfrowanie.
- Nieustanna ewolucja napędzana nowym sprzętem, aplikacjami i wymaganiami użytkowników
- Techniki konstruowania bardzo złożonych systemów
- Projektowanie i programowanie systemów operacyjnych to szereg ciekawych wyzwań!

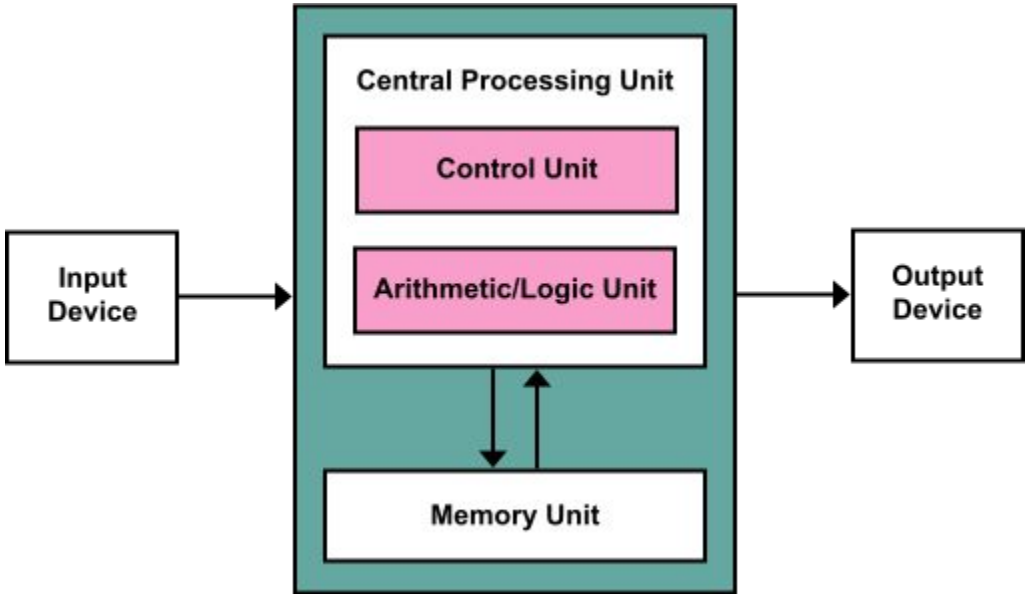
Formuła zajęć

1. Ćwiczenia
   a. system deklaracji
   b. ~8 zadań na listę
   c. łącznie ~12 list ćwiczeniowych
2. Pracownie
   a. procesy i wątki
   b. synchronizacja i komunikacja
3. Projekty programistyczne
   a. menadżer pamięci (typu `malloc` / `free`)
   b. system plików do odczytu (`FUSE` i `FAT` lub `ext2`)

Do zadań programistycznych będą dostarczone szkielety rozwiązań → pozostaje implementacja ciekawych rzeczy!

Czym jest system operacyjny?

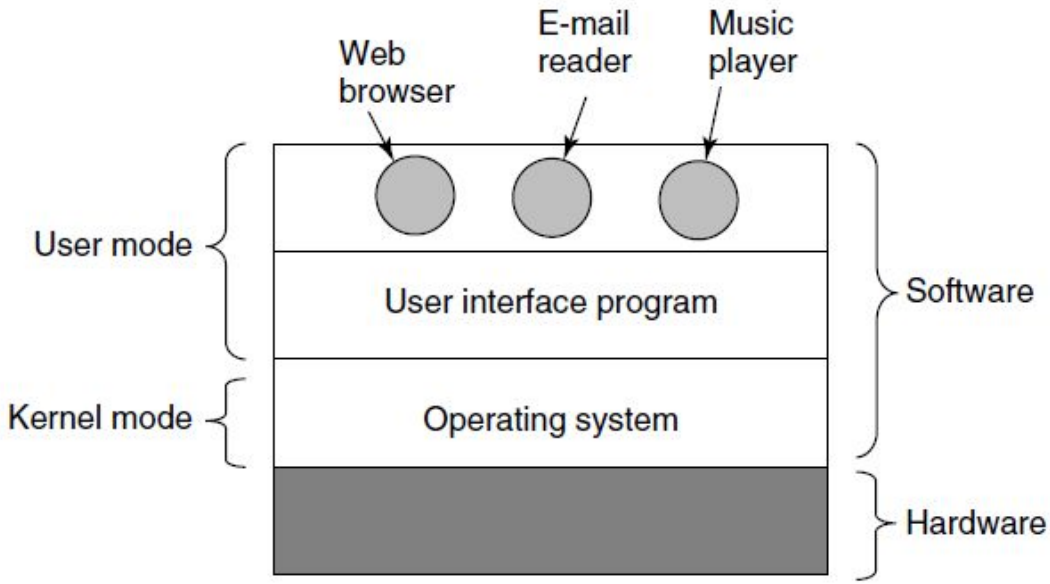
Co robią komputery?



1. Przetwarzają dane
2. Komunikują się
3. Przechowują dane

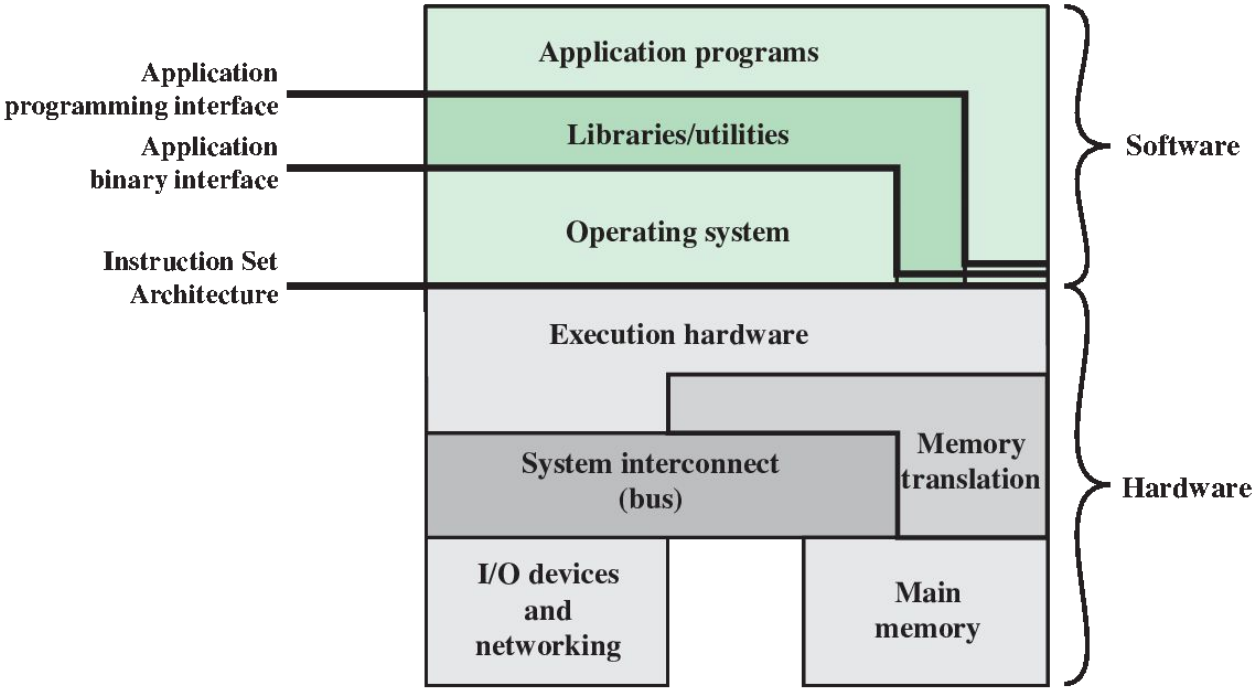
Kluczowe funkcje systemu operacyjnego

- Kontroluje zasoby maszyny
- Sprawiedliwie udostępnia zasoby programom
- Dostarcza środowiska uruchomieniowego dla programów
- Nadzoruje wykonanie programów (izolacja)



Składowe systemu operacyjnego

- Powłoka graficzna lub/i tekstowa
- Biblioteki dzielone
- Narzędzia systemowe
- Sterowniki
- Jądro



Ważne funkcje systemu operacyjnego

- Kluczowa infrastruktura: kompilatory, edytor, powłoka
- Zarządzanie systemem: rozruch, zamknięcie, nadzór
- Abstrakcje i usługi:
  - API: procesy, wątki, komunikacja, pliki, model aplikacji
  - Dzielenie zasobów: planowanie, wirtualizacja, multipleksowanie
  - I/O: sterowniki, lokalne / rozproszone systemy plików, stos sieciowy
  - Bezpieczeństwo: uwierzytelnianie, szyfrowanie, uprawnienia, audyt
  - Lokalny i zdalny dostęp: konsola tekstowa lub graficzna, SSH
- Biblioteki: matematyczna, gniazda sieciowe, RPC, kryptografia, interfejs użytkownika, multimedia
- Inne: dziennikowanie, odpluskwianie, profilowanie, śledzenie

Jądro vs. system operacyjny

Wykonywane w trybie uprzywilejowanym. Dostarcza podstawowego (?) zestawu usług.

Czy używając tego samego jądra da się zrobić różne SO?

Android vs. Ubuntu:

- inne biblioteki i narzędzia systemowe
- inna architektura systemu
- inny model dystrybucji oprogramowania
- to samo jądro (prawie) Linux

OpenDarwin vs. macOS

System operacyjny vs. dystrybucja

Rozproszony czy scentralizowany model dystrybucji oprogramowania?

Debian GNU/Linux vs. Fedora

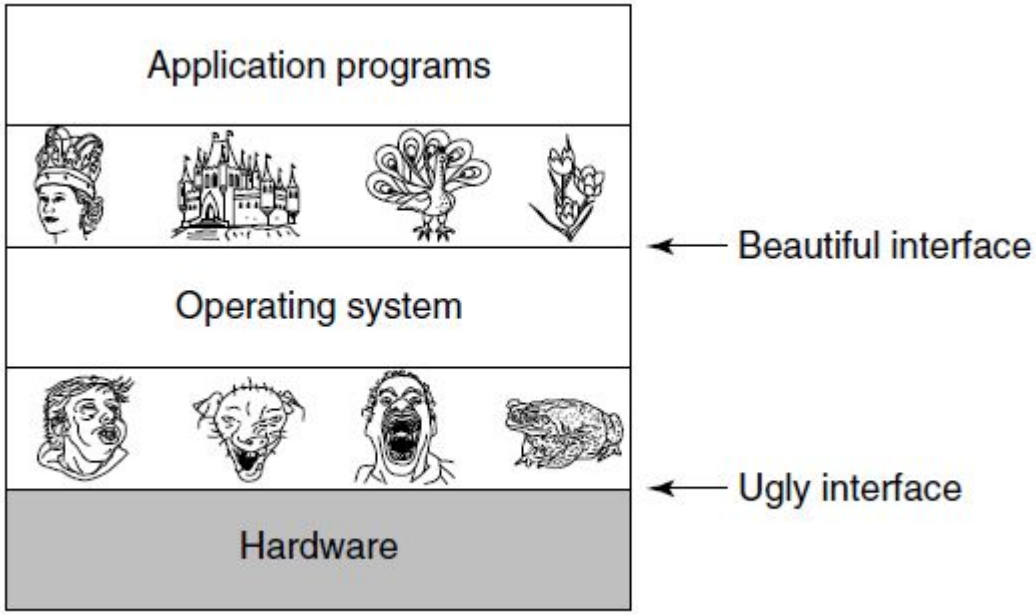
- Linux to samo jądro!
- ten sam zestaw narzędzi systemowych i bibliotek
- inny system pakietów

Czy Windows 10 i macOS są dystrybucją?

Rozszerzona maszyna

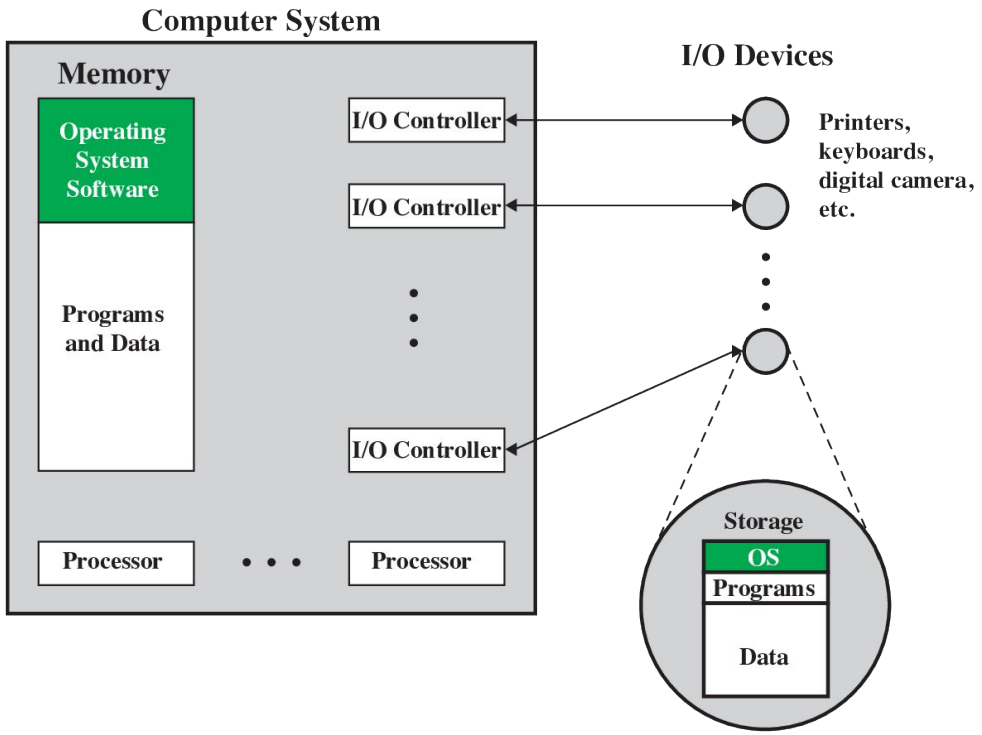
Interfejs sprzętowy jest skomplikowany, niewygodny i zorganizowany na przeróżne sposoby.

SO organizuje środowisko uruchomieniowe – maszyna "wirtualna" z abstrakcyjnym zunifikowanym interfejsem.



Jakimi zasobami zarządza SO?

- pamięcią
  - pamięć fizyczna
  - tablica stron
  - przestrzeń adresowa
- procesorem
  - kontekst i tryb pracy
  - czas (przełączanie)
  - translacja adresów
  - przerwania
  - inne (cache, TLB)
- urządzeniami wejścia-wyjścia
- ….



Historia systemów operacyjnych

Jaką drogę muszą przebyć nowe SO?

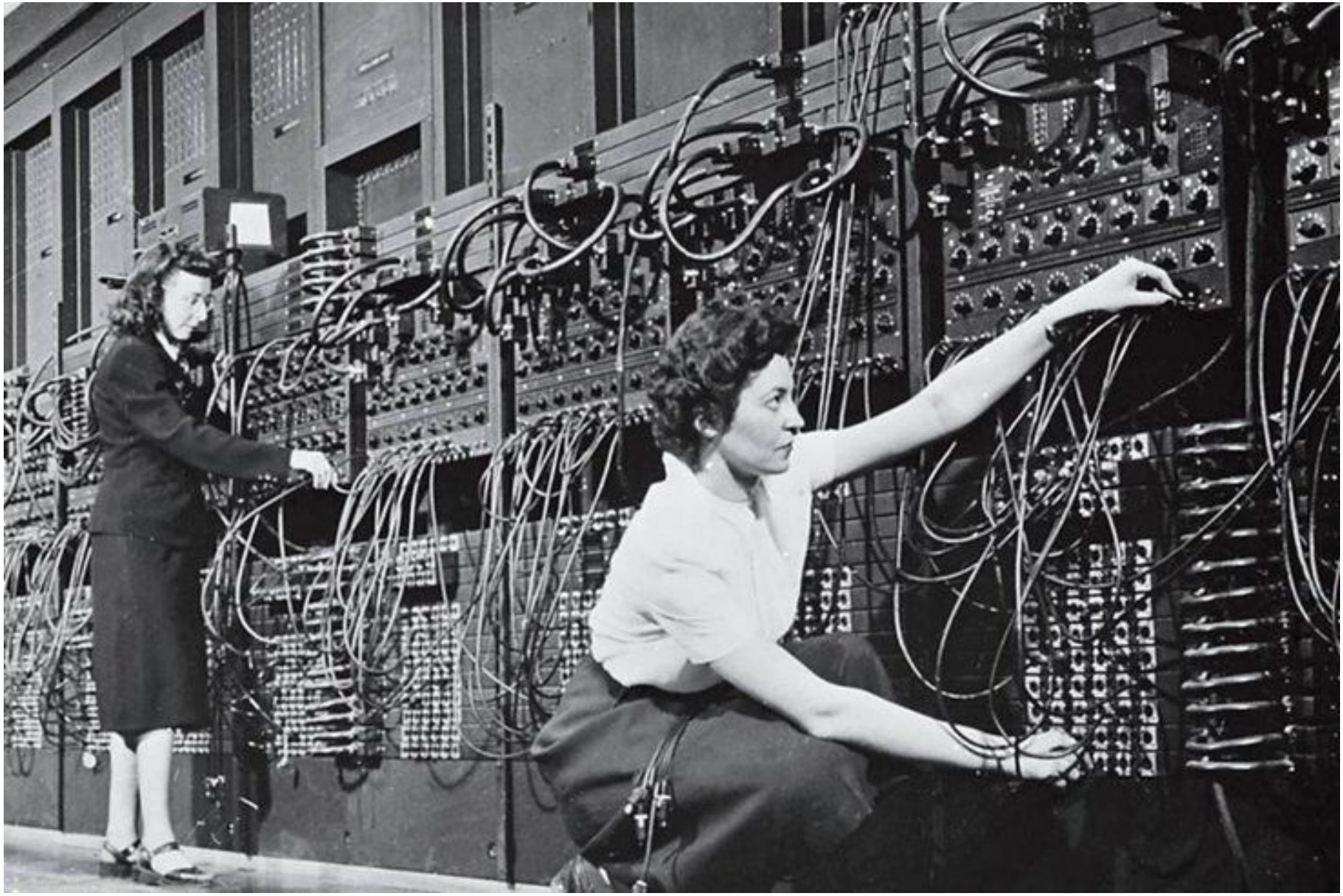
"Ontogeneza jest rekapitulacją filogenezy" – Ernst Haeckel

Ontogeneza – rozwój embrionu.
Filogeneza – rozwój gatunków.

Zmieniają się warunki środowiskowe (sprzęt, języki programowania, wymagania operacyjne): pewne idee przetrwały, inne uzyskały miano przestarzałych.

Powroty z zaświatów się zdarzają: Magnetic Tape Data Storage Breakthrough Will Make Your Hard Drive Seem Tiny.

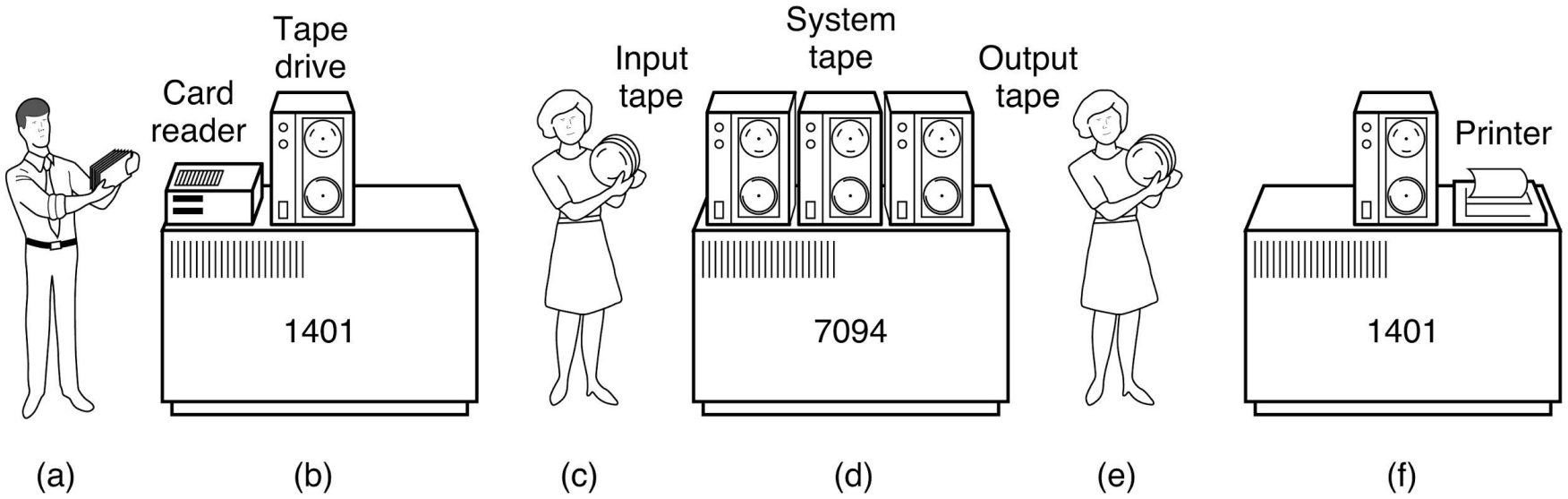
(Tak, Google też używa taśm magnetycznych do backupów!)



Komputery mainframe

Drogie w utrzymaniu. Planowanie zadań na kartce. Ręczna obsługa. Duże zużycie prądu. Jak nie marnować pieniędzy?

Pierwszym rozwiązaniem: systemy wsadowe.




IBM 526
PRINTING SUMMARY PUNCH

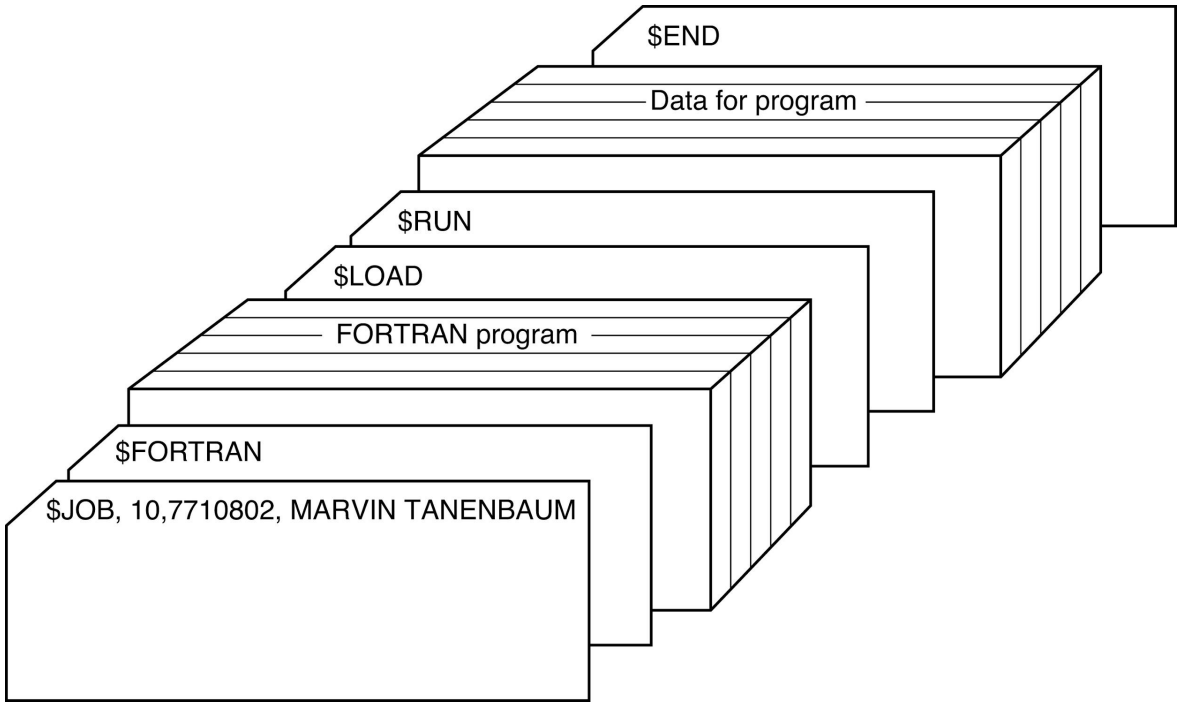

Card Reader / Punch
IBM 1402

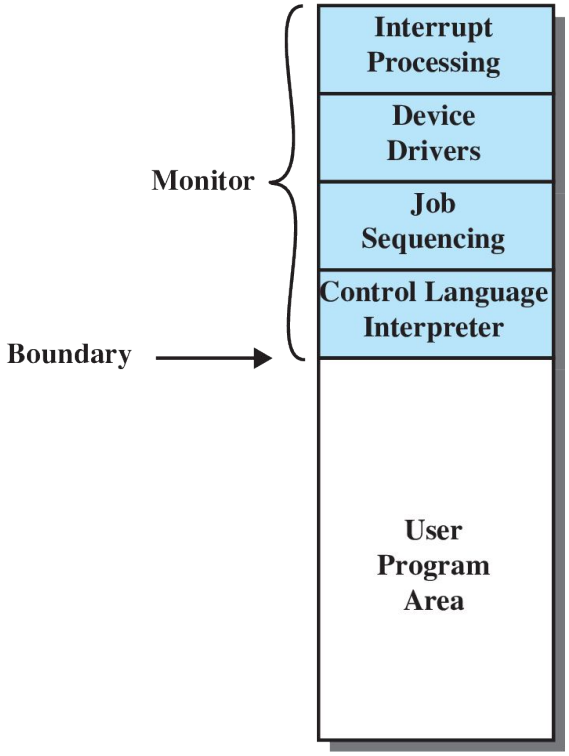

IBM 3481
IBM 3481
IBM 3481


IBM
1401
IBM
1402
CARD READ PUNCH

Systemy wsadowe: lata ‘50 (1)

Jak bezpiecznie i wygodnie uruchamiać zadania? Co zrobić jeśli zadanie się zawiesiło lub zakończyło błędem?





Stack of 62,500 punched cards 5 MB worth, holding the control program for the giant SAGE military computer network. 1955

Systemy wsadowe: lata ‘50 (2)

Wymagane wsparcie sprzętowe do realizacji bezpiecznego systemu wsadowego:

- ochrona pamięci monitora
- czasomierz
- przerwania
- tryby pracy procesora
- instrukcje uprzywilejowane
- wywołania systemowe

Wieloprogramowość

Do systemu wchodzą zadania ograniczone procesorem lub wejściem-wyjściem. Jak zwiększyć stopień wykorzystania?

Zamrozić zadania, które nie mogą uzyskać dostępu do danego zasobu! Uruchomić te, które mogą postępować z obliczeniami.

Potrzebne bardziej złożone zarządzanie pamięcią. Wiele programów w pamięci na raz. Wymiana do pamięci zewnętrznej.

Lepiej, ale nadal brak interakcji z maszyną.


TELETYPE
SHIFT

Systemy z dzieleniem czasu: minikomputery lat ‘60

Brak interakcji uniemożliwiał iteracyjne rozwiązywanie problemów. Co gdyby dodać dalekopis (ang. *teletype*)?

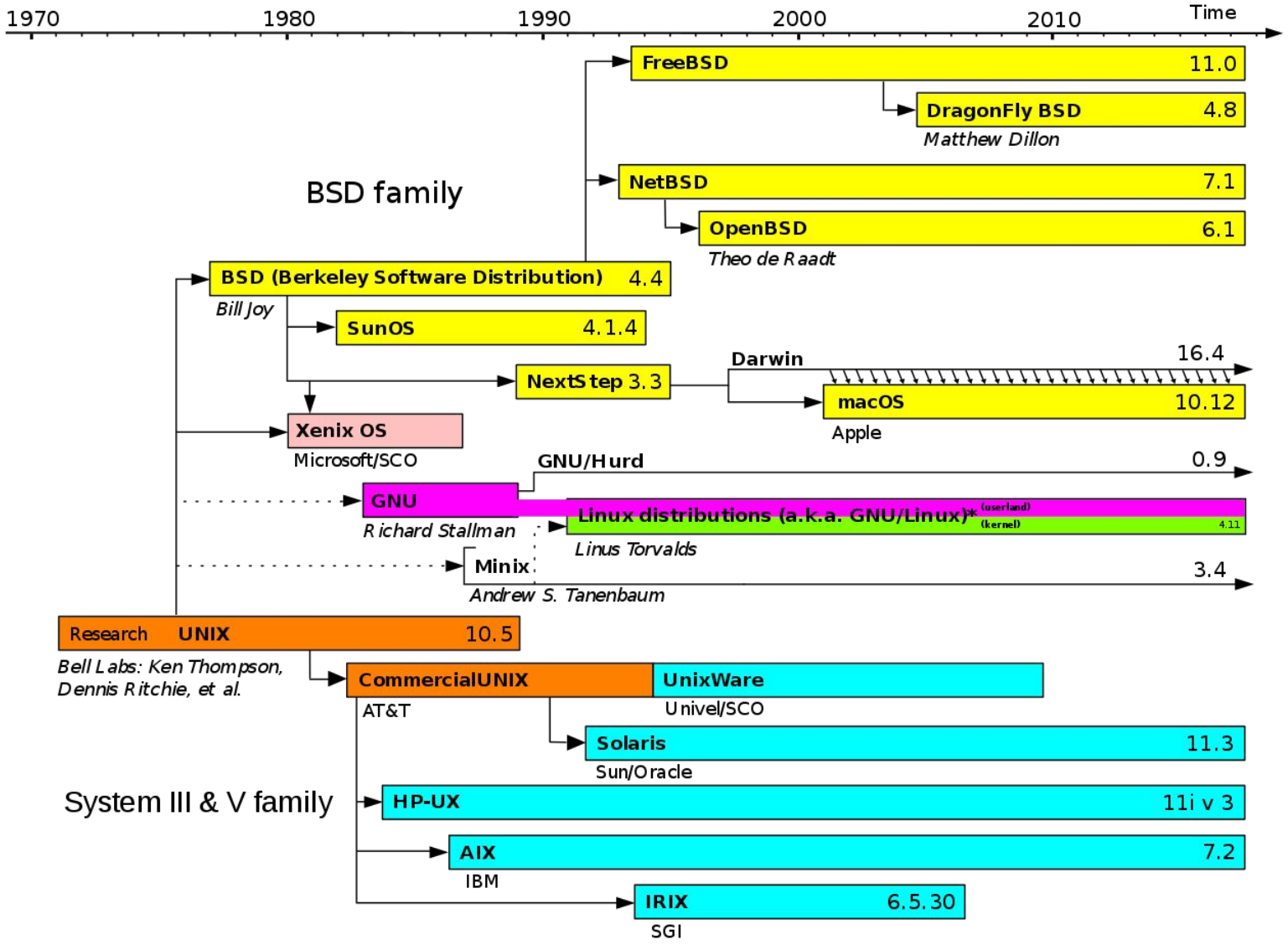
Dzielimy czas procesora na kwanty. Przełączamy zadania wystarczająco szybko, żeby użytkownik się nie zorientował.

Nowe techniki:

- buforowanie wejścia-wyjścia (edycja linii)
- planista krótkoterminowy
- wywłaszczanie zadań
- stronicowanie

PDP-11/40: na takim komputerze powstał UNIX

Początek ery uniksowej ‘70



Filozofia uniksowa

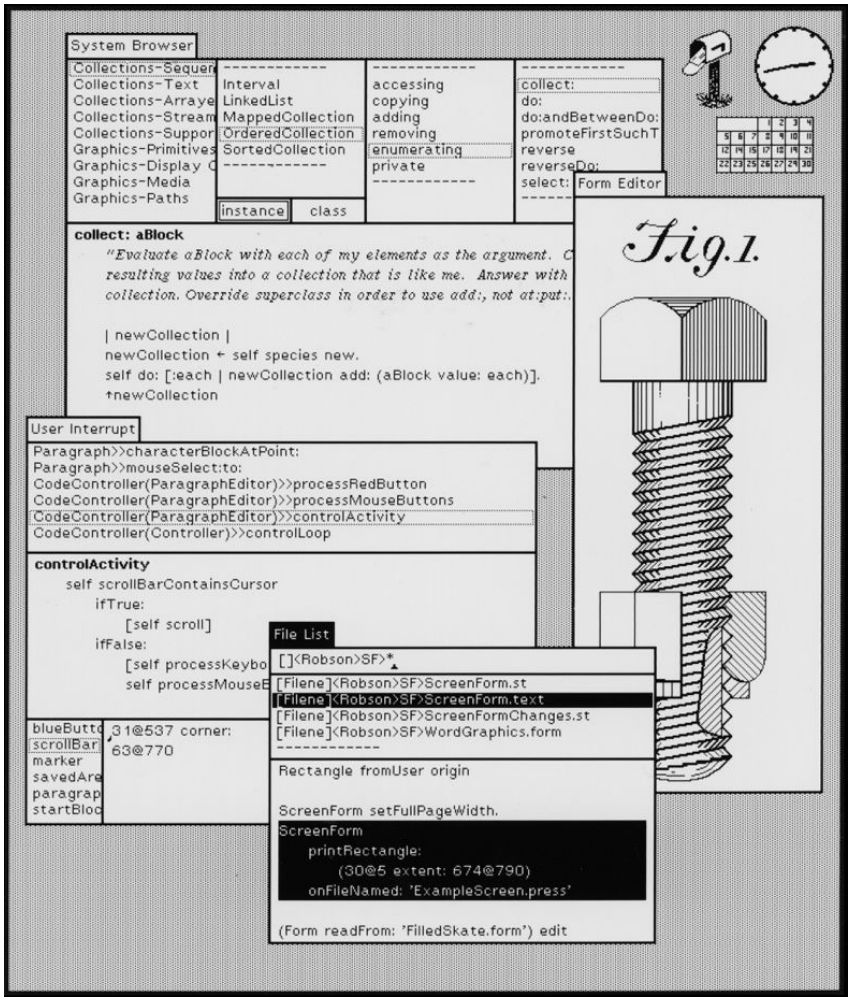
Unix powstawał w kulturze hakerów (Bell Labs) na początku ‘70.

- Small is beautiful.
- Make each program do one thing well.
- Build a prototype as soon as possible.
- Choose portability over efficiency.
- Store data in flat text files.
- Use software leverage to your advantage.
- Use shell scripts to increase leverage and portability.
- Avoid captive user interfaces.
- Make every program a filter.

– The UNIX Philosophy, Mike Gancarz

The Mother of All Demos – 9 grudzień 1968

Xerox Alto: prekursor systemów okienkowych ‘73



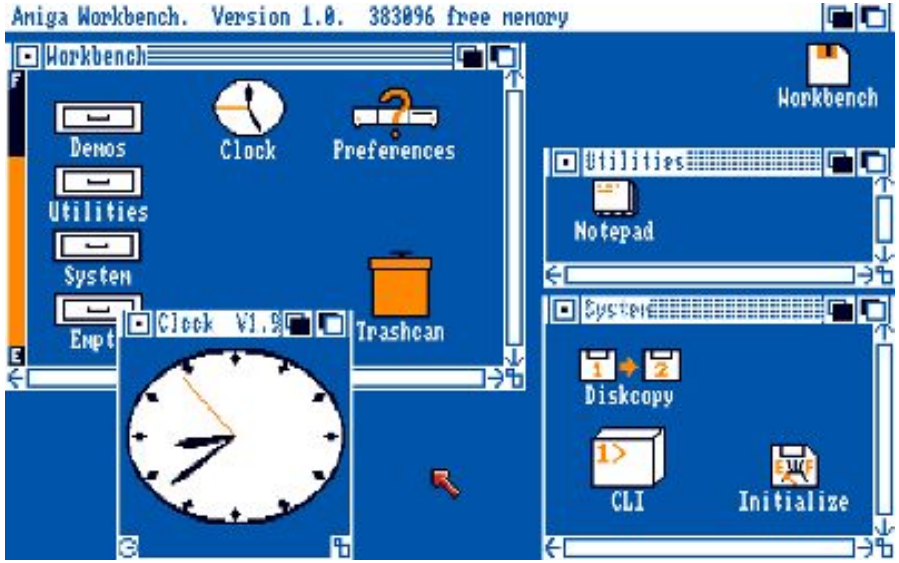
Pierwszy pakiet w Internecie (październik 1969)

Lo and Behold, Reveries of the Connected World (Werner Herzog, 2016)

Era mikrokomputerów – ‘77 do dziś

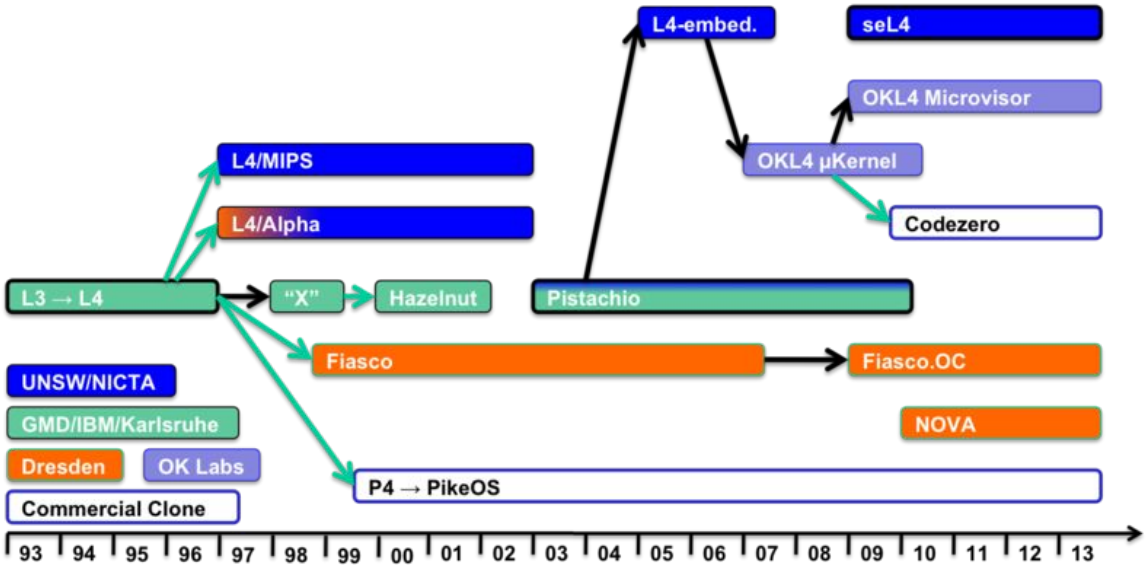
Systemy operacyjne dla mikrokomputerów

- CP/M → DOS → FreeDOS
- MS DOS → Win 1.x – 3.x → Win 9x
- PC DOS → OS/2 → eComStation
- RiscOS (Acorn Archimedes)
- MacOS (do wersji 9)
- VMS → Windows NT
- AmigaOS → AROS
- Atari TOS → FreeMiNT
- BeOS → Haiku



Inne ważne systemy operacyjne

- Plan9 → Inferno
- rodzina mikrojąder L4
- Minix
- Cisco IOS
- VxWorks



Typy systemów operacyjnych

Mogą znacząco różnić się udostępnionymi abstrakcjami i wymaganym wsparciem sprzętowym:

- serwerowe
- dla komputerów osobistych
- czasu rzeczywistego
- wbudowane
- rozproszone
- sieciowe
- monitory maszyn wirtualnych

Pytania?