Systemy operacyjne

Mariusz Żynel
mariusz@math.uwb.edu.pl
http://math.uwb.edu.pl/~mariusz/

Uniwersytet w Białymstoku

2024/2025

Co to jest system operacyjny, do czego służy i jak działa?

Dlaczego musi być egzamin?

Egzamin

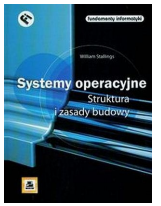
- Warunkiem dopuszczenia do egzaminu jest uzyskanie zaliczenia ćwiczeń na ocenę pozytywną
- Egzamin pisemny składa się z 50 pytań w formie testu
- Do każdego pytania są 3 propozycje odpowiedzi, ale tylko jedna jest poprawna
- Odpowiedź na każde z pytań testowych warta jest 0 lub 1
- Punktacja:
  - 5.0 : 46
  - 4.5 : 41
  - 4.0 : 36
  - 3.5 : 31
  - 3.0 : 26
- Obowiązuje materiał z wykładu:
  http://math.uwb.edu.pl/~mariusz/

Plan wykładu

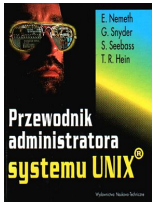
- Trochę historii, ale nie za dużo
- Powłoka, praca w konsoli, znaki specjalne
- Strumienie, potoki i rurociągi
- Użytkownicy, grupy, `root` i jak nim zostać
- Systemy plików, prawa dostępu, dowiązania i lepkie bity
- Wyrażenia regularne
- Przegląd narzędzi
- Pisanie skryptów
- Procesy i zadania powłoki, komunikacja między procesami
- Kolejkowanie, szeregowanie i przydzielanie zasobów
- Gdzie się podział `autoexec.bat`, czyli start systemu
- Dyski, macierze, wolumeny i montowanie zasobów dyskowych
- Podstawowe usługi
- Logi, planowanie zadań, czyli ciężka praca administratora

Literatura

- Stallings W.
  Systemy operacyjne: struktura i zasady budowy
  Wydawnictwo Naukowe PWN, Warszawa 2006.

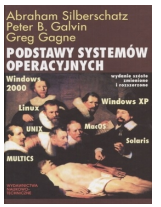


- Nemeth E. i in.
  Przewodnik administratora systemu Unix
  WNT, Warszawa 1998.

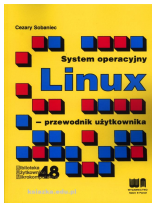


- Frisch A.
  Unix. Administracja systemu
  Wyd. 3, ReadMe/O'Reilly, 2003.

- Silberschatz A., Galwin P.B., Gagne G.
  Podstawy systemów operacyjnych
  WNT, Warszawa 2005.



- Sobaniec C.
  System operacyjny Linux - przewodnik użytkownika
  Nakom, Poznań 2002.



Co to jest epoch date i dlaczego to takie ważne?

Po co nam w ogóle system operacyjny?

- Przetwarzanie wsadowe (batch processing) [video]
  - Jeden użytkownik, jedno zadanie w danym czasie
  - Optymalizacja użycia komputera i likwidacja przestojów
  - Kolejkowanie zadań
- Podział czasu (time-sharing)
  - Wielu użytkowników, wiele zadań w tym samym czasie
  - Jeden użytkownik nieefektywnie gospodaruje komputerem, ale przy wielu równoległych użytkownikach przerwy w pracy jednego wypełnione są aktywnością pozostałych
  - Przydział zasobów procesora, pamięci oraz urządzeń `I/O` na określony czas
  - Wywłaszczanie
  - Większe wymagania co do szybkości procesora i rozmiarów pamięci

RTOS – system operacyjny czasu rzeczywistego

- Deterministyczny, przewidywalny
  - Soft – większość zadań ma określony najdłuższy czas wykonania
  - Hard – każde zadanie ma określony najdłuższy czas wykonania
- Przełączanie zadań następuje w oparciu o zdarzenia a nie przerwania zegara
- Nieistotna jest wydajność i przepustowość, lecz spełnianie narzuconych wymagań czasowych
- Bezwzględne przestrzeganie priorytetów zadań podczas ich realizacji
- Przerwania sprzętowe i wywołania systemowe muszą być obsłużone w ściśle określonym czasie
- Większe znaczenie ma jak szybko i przewidywalnie reaguje system niż jak dużo może zrobić w określonym czasie
- Przykłady: QNX, FreeRTOS, RT-Linux, VxWorks, OS-9

Pierwsze systemy z podziałem czasu

- 1961 – Compatible Time-Sharing System (CTSS)
  - System operacyjny ogólnego przeznaczenia
  - Prototypowy shell, e-mail, edytor QED, BASIC
  - Pierwszy system z logowaniem poprzez hasło
  - MIT, działał na IBM 7090

- 1961 – PLATO II
  - Rozproszony system edukacyjny
  - Prototypowe forum, message board, chat rooms, instant messaging, multiplayer video game
  - UI, działał na ILLIAC I oraz CDC 1604

- 1964 – Dartmouth Time-Sharing System (DTSS)
  - BASIC, pierwsze IDE, IPC w formie potoków
  - 300 użytkowników jednocześnie z USA, Kanady i Europy, sukces komercyjny
  - DC, działał na GE-235 oraz GE-635

Burzliwe lata 60-te XX-wieku

- 1964 – w Bell Labs przy współpracy z GE i MIT powstaje projekt innowacyjnego systemu operacyjnego z podziałem czasu o nazwie Multics (Multiplexed Information and Computing Service)
  - Kontynuacja CTSS
  - Pisany w języku PL/I, działał na GE-635 oraz GE-645
- 1969 – Ken Thompson tworzy grę komputerową Space Travel
  - Symulacja podróży po Układzie Słonecznym
  - Napisana i uruchomiona na Multics
- 1969 – prace nad Multics zostają zawieszone
  - Założenia są nowatorskie, ale system jest zbyt skomplikowany
  - Ogromne koszty projektu
- 1969 – gra Space Travel zostaje przeniesiona an GECOS
  - Przepisanie kodu gry na Fortran
  - GECOS działa w trybie wsadowym na GE-635
- 1969 – Thompson zaczyna przenosić swoją grę na DEC PDP-7

Minikomputer DEC PDP-7

Czy fotel jest integralnym komponentem systemu?

Początki systemu Unix

- 1970 – Ken Thompson tworzy nowy assembler oraz całkowicie nowy, prosty, jednozadaniowy system operacyjny w celu uruchomienia swojej gry Space Travel na PDP-7
- 1970 – Projektem Thompsona interesują się Dennis Ritchie, Brian Kernighan, Douglas McIlroy i Joe Ossanna współautorzy Multics, powstaje system operacyjny Unics (Uniplexed Information and Computing Service), którego nazwa zaczyna być wymawiana jako Unix
- 1971 – na potrzeby swego systemu operacyjnego Ken Thompson opracowuje nowy język B oparty na BCPL
- 1971/1972 – Ken Thompson i Dennis Ritchie rozwijają język B, powstaje NB, a potem C
- 1972 – przepisanie Unixa z assemblera na C (portability)

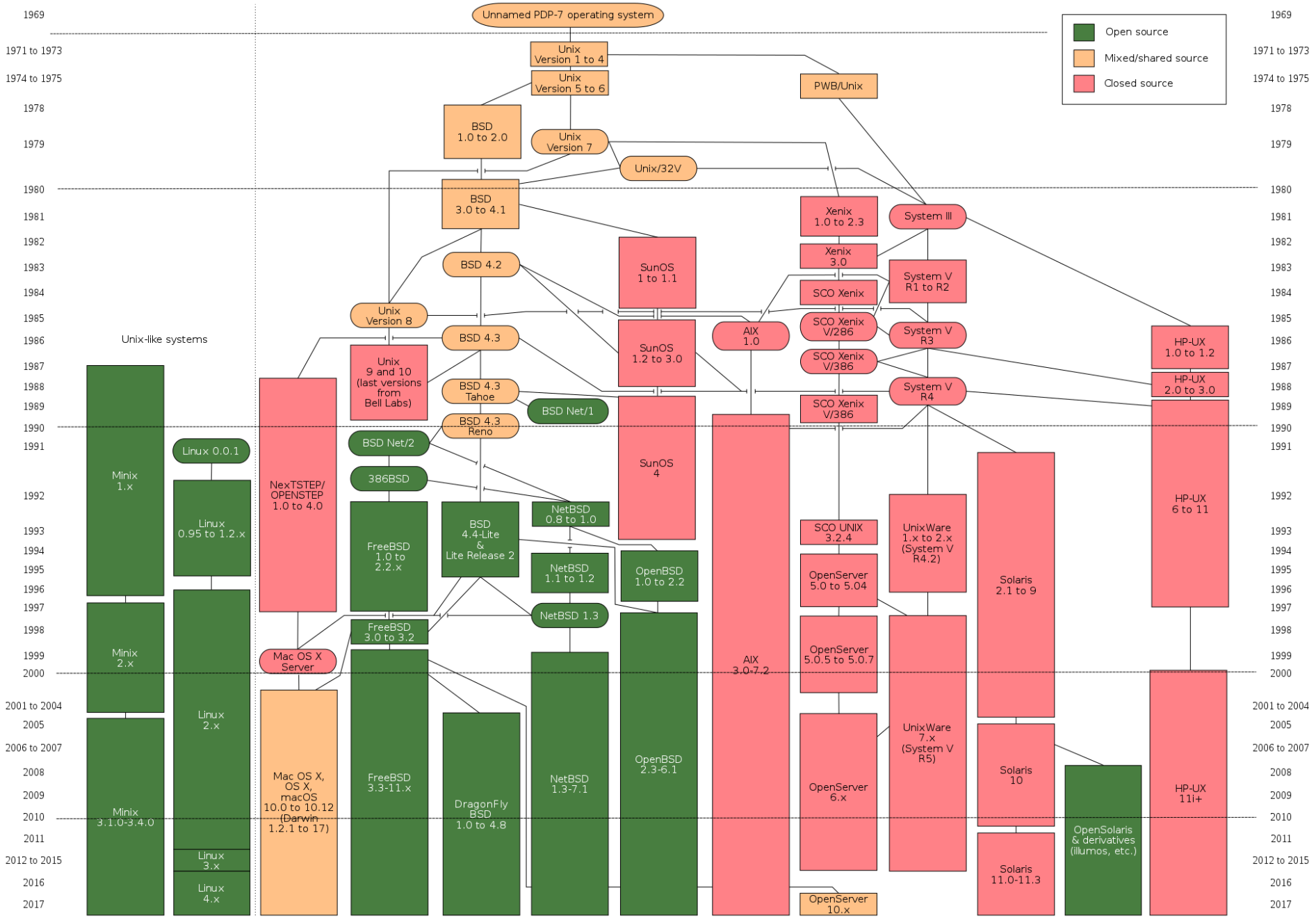
Epoch date

- Pierwsze wersje Unixa na początku lat 70-tych odliczają czas z częstotliwością 60Hz
- 32-bitowa liczba całkowita bez znaku może reprezentować czas w zakresie do 829 dni
- Epoch date ustalono na 1971-01-01 00:00:00
- Nowsze wersje Unixa odliczają czas z częstotliwością 1Hz
- 32-bitowa liczba całkowita bez znaku może reprezentować czas w zakresie do około 136 lat
- Epoch date ustalono na 1970-01-01 00:00:00
- Maksymalna data reprezentowana przez 32-bitową liczbę całkowitą ze znakiem to 2038-01-19, potem data zostanie zmieniona na 1901-12-13.

Co dalej z tym Unixem?

Dalsze losy Unixa

- 1975 – pierwsza sprzedana licencja Unixa do University of Illinois
- 1979 – powstaje BSD i Unix Version 7
- 1983 – powstaje SunOS oraz System V
- 1987 – Andrew Tanenbaum tworzy system MINIX
- 1991 – Linus Torvalds tworzy system Linux, powstaje Solaris
- 2005 – powstaje Solaris 10 i OpenSolaris
- 2010 – nabycie Sun Microsystems przez Oracle


Open source
Mixed/shared source
Closed source
Unix-like systems
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
Unix Version 7
PWB/Unix
BSD 1.0 to 2.0
Unix/32V
BSD 3.0 to 4.1
BSD 4.2
BSD 4.3
BSD 4.3 Tahoe
BSD 4.3 Reno
Unix Version 8
Unix 9 and 10 (last versions from Bell Labs)
BSD Net/1
BSD Net/2
386BSD
FreeBSD 1.0 to 2.2.x
FreeBSD 3.0 to 3.2
FreeBSD 3.3-11.x
BSD 4.4-Lite & Lite Release 2
NetBSD 0.8 to 1.0
NetBSD 1.1 to 1.2
NetBSD 1.3
NetBSD 1.3-7.1
DragonFly BSD 1.0 to 4.8
SunOS 1 to 1.1
SunOS 1.2 to 3.0
SunOS 4
OpenBSD 1.0 to 2.2
OpenBSD 2.3-6.1
AIX 1.0
AIX 3.0-7.2
Xenix 1.0 to 2.3
Xenix 3.0
SCO Xenix
SCO Xenix V/286
SCO Xenix V/386
SCO Xenix V/386
SCO UNIX 3.2.4
OpenServer 5.0 to 5.04
OpenServer 5.0.5 to 5.0.7
OpenServer 6.x
OpenServer 10.x
System III
System V R1 to R2
System V R3
System V R4
UnixWare 1.x to 2.x (System V R4.2)
UnixWare 7.x (System V R5)
Solaris 2.1 to 9
Solaris 10
Solaris 11.0-11.3
OpenSolaris & derivatives (illumos, etc.)
HP-UX 1.0 to 1.2
HP-UX 2.0 to 3.0
HP-UX 6 to 11
HP-UX 11i+
Minix 1.x
Minix 2.x
Minix 3.1.0-3.4.0
Linux 0.0.1
Linux 0.95 to 1.2.x
Linux 2.x
Linux 3.x
Linux 4.x
NexTSTEP/ OPENSTEP 1.0 to 4.0
Mac OS X Server
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)

Dlaczego Unix?

Podstawowe cechy charaketeryzujące system Unix

- Wielozadaniowość – wiele zadań w tym samym czasie
- Wielodostępność – wielu użytkowników, praca lokalna i zdalna
- Wieloplatformowość – od tabletu i smartfona po mainframe

Filozofia systemu UNIX

Dwie proste zasady:

- Wszystko jest plikiem
- Keep it small and simple

The power of a system comes more from the relationships among programs than from the programs themselves

Brian Kernighan, Rob Pike

To co odróżnia Unix od innych systemów:

- dane przechowywane są w plikach tekstowych
- hierarchiczny system plików
- traktowanie urządzeń jak plików
- traktowanie części komunikacji międzyprocesowej jak plików
- wiele małych programów narzędziwych, które można używać łącznie z linii poleceń w potokach, zamiast pojedynczych, monolitycznych programów, które robią wszystko

Dlaczego UNIX jest taki nudny?

- Powtarzalność – z uporem maniaka wykonuje to co mu każesz, za każdym razem w ten sam sposób, bez komentarzy i złośliwych uwag
- Przewidywalność – jeśli nie jesteś pewien jak zareaguje system to zajrzyj do dokumentacji albo do kodu źródłowego

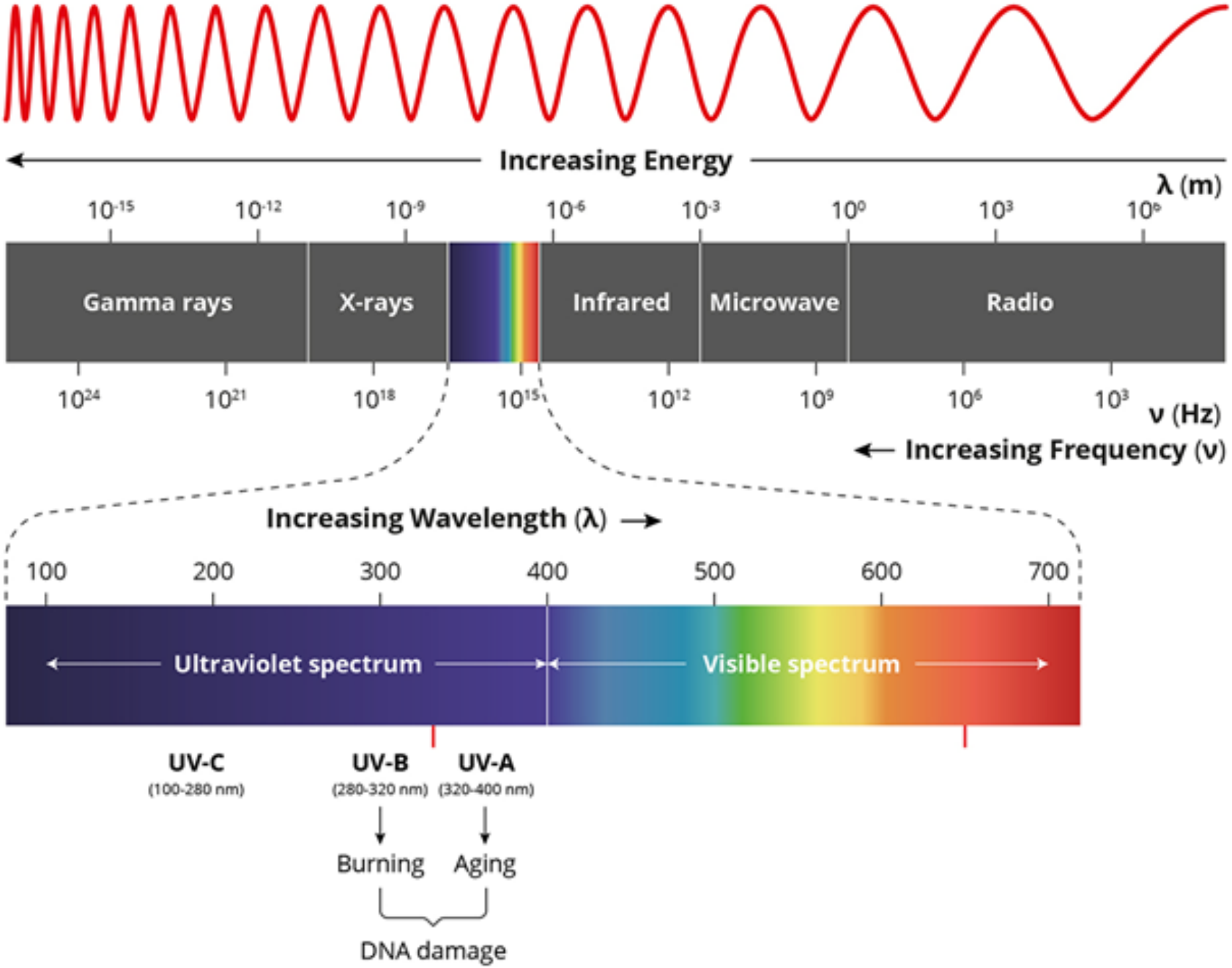
Dlaczego UNIX jest taki nudny?

- Powtarzalność – z uporem maniaka wykonuje to co mu każesz, za każdym razem w ten sam sposób, bez komentarzy i złośliwych uwag
- Przewidywalność – jeśli nie jesteś pewien jak zareaguje system to zajrzyj do dokumentacji albo do kodu źródłowego

Dlaczego UNIX jest taki nudny?

- Powtarzalność – z uporem maniaka wykonuje to co mu każesz, za każdym razem w ten sam sposób, bez komentarzy i złośliwych uwag
- Przewidywalność – jeśli nie jesteś pewien jak zareaguje system to zajrzyj do dokumentacji albo do kodu źródłowego

Dlaczego niebo jest niebieskie i co to ma wspólnego z transmisją danych?


Increasing Energy
λ (m)
10-15
10-12
10-9
10-6
10-3
100
103
106
Gamma rays
X-rays
Infrared
Microwave
Radio
1024
1021
1018
1015
1012
109
106
103
ν (Hz)
Increasing Frequency (ν)
Increasing Wavelength (λ)
100
200
300
400
500
600
700
Ultraviolet spectrum
Visible spectrum
UV-C
(100-280 nm)
UV-B
(280-320 nm)
UV-A
(320-400 nm)
Burning
Aging
DNA damage

```
> FIN < ACK < FIN > ACK

init 5
```