Unix – przegląd

Paweł Józiak

Wydział Matematyki i Informatyki Politechniki Warszawskiej

10 X 2022

Plan zajęć

1. FLOSS.
2. Zadania i komponenty systemu operacyjnego.
3. POSIX, SUS a Unix
4. Systemy Unix, Unix-based, Unix-like
5. Ideologia Unix

Plan zajęć

1. FLOSS.
2. Zadania i komponenty systemu operacyjnego.
3. POSIX, SUS a Unix
4. Systemy Unix, Unix-based, Unix-like
5. Ideologia Unix

Free Libre Open Source Software

- Zagadnienie *wolności* oprogramowania tyczy się wolnego dostępu do:
  - używania,
  - analizowania,
  - modyfikacji,
  - rozpowszechniania

  oprogramowania, dzięki udostępnieniu jego kodu źrodłowego (*source code*).
- FLOSS rozdziela zagadnienia własność od autorstwa oprogramowania:
  - rozpowszechnianie oryginalnego bądź zmodyfikowanego oprogramowania jest możliwe wyłącznie jeśli informacja o źródle, oryginalnym autorze i licencji jest składnikiem rozpowszechnianego pakietu
  - oprogramowanie *należy* do każdego człowieka.

FLOSS a matematyka

- Stwierdzenia matematyczne są opatrywane referencjami autorów, zaś nie bardzo jest sens mówić o tym, by były czyjąś własnością
- Każdy może:
  - używać
  - dowodzić (analizować)
  - uogólniać (modyfikować)
  - uczyć (rozpowszechniać)

  twierdzenia matematyczne. Nie znaczy to jednak, że w ten sposób ktokolwiek bierze w posiadanie jakikolwiek fragment matematyki.

Własnościowe oprogramowanie a laboratoria

- Studenci mogą nieodpłatnie korzystać z komputerów na wydziale,
- Komputery wciąż pozostają własnością wydziału:
  - nie zezwala się na żadne nieuzgodnione modyfikacje sprzętu bądź oprogramowania w komputerach wydziałowych;
  - studenci nie mogą tych komputerów sprzedać, ani odpładnie użyczyć, osobom trzecim;
  - picie i jedzenie jest zabronione w trakcie korzystania z komputerów w laboratoriach;
  - studenci nie mogą ich używać do celów zarobkowych;
  - Wydział pozostawia sobie prawo do zmiany warunków użytkowania komputerów, wraz z wycofaniem dotychczas przyznanych uprawnień.

FLOSS: wolność

- zwrot *free* w nazwie FLOSS wyraża aspekty etyczno-prawne (*freedom*), nie ekonomiczne (*free of charge*),
- *Freeware, demo, adware, trial*, a czasem również *shareware* dotyczą możliwości nieodpłatnego używania oprogramowania własnościowego. Autorzy/dystryburorzy oprogramowania są właścicielami oprogramowania, użytkownicy kupują bądź dostają nieodpłatnie prawo do wykorzystania oprogramowania. Właściciele mogą ustalać (i później zmieniać) zasady, na jakich nastąpiło użyczenie.
- FLOSS nie implikuje, że oprogramowanie jest nieodpłatne.

Free Libre vs. Open Source

- FLOSS jest ogólnym terminem obejmującym zarówno Wolne Oprogramowanie (*Free Software*), jak również Otwarte Oprogramowanie (*Open Source Software*).
- Ten pierwszy termin dotyka aspektów etycznych i prawnych, które są dane użytkownikowi, zaś drugie dotyka aspektów technicznych.
- Wiele licencji typu FLOSS jest dostępnych na świecie, choć źródłowo większość z nich wywodzi się od licencji Wolnego Oprogramowania GNU GPL albo Otwartego Oprogramowania BSD.

Licencja Berkeley Software Distribution

- Licencja BSD orzeka, że:
  - informacja o autorach oraz licencji musi być uwzględniona w oryginalnym oraz zmodyfikowanym kodzie oraz plikach binarnych, oraz materiałach dołączonych (dokumentacji etc)
  - Nazwiska autorów nie mogą być używane dla reklamy i promocji oprogramowanie wywiedzionego z oprogramowania licencjonowanego BSD bez uprzedniej pisemnej zgody tychże,
  - oprogramowanie jest dostarczone takie, jakim jest (*as is*), a żadne wyjątki i ograniczenia nie mogą być powodem roszczeń.
- Oprogramowanie BSD może być używane przez aplikacje o licencji własnościowej.

Licencja GNU General Public License

- Licencja GNU GPL gwarantuje prawa autora, a wolności użytkownika nie mogą ich naruszać:
  - każdy licencjobiorca, który podlega tej licencji, ma prawo do modyfikowania oprogramowania, a także kopiowania i rozpowszechniania pracy bądź oprogramowania z niej wywiedzionego. Licencjobiorca ma prawo do ustanowienia opłaty za swoje usługi, bądź do ustalenia stawki na poziomie 0.
  - Oprogramowanie może być rozpowszechniane jedynie w postaci Otwartego Oprogramowania pod licencją GNU GPL,
  - Żadne ograniczenie praw nadanych przez licencję GNU GPL nie może być zapostulowane ani do oryginalnego, ani do wywiedzionego oprogramowania.
- Oprogramowanie GNU GPL nie może być związane z oprogramowaniem własnościowym. Istnieje licencja *GNU Leser General Public Licence*, która pozwala na współdzielenie oprogramowania GNU jako bibliotek zależnych w oprogramowaniu własnościowym.

FLOSS a koszty

- Zagadnienie oprogramowania FLOSS dotyczy aspektów prawno-etycznych, nie ekonomicznych. Licencje FLOSS pozwalają na zarabianiu na:
  - rozpowszechnianiu nośników danych (CD, DVD etc),
  - usługach szkoleniowych,
  - pomocy z instalacji/konfiguracji,
  - wszelkiej postaci wsparcia,
  - gwarancji zgodności ze sprzętem,
  - dostosowywanie oprogramowania do potrzeb klienta.
  - ...
- CentOS oraz RedHat Enterprise Linux to zasadniczo te same systemy. CentOS jest dostępny nieodpłatnie, zaś Red Hat Enterprise Linux jest sprzedawany wraz z pełnym pakietem obsługi wdrożeniowej.

Plan zajęć

1. FLOSS.
2. Zadania i komponenty systemu operacyjnego.
3. POSIX, SUS a Unix
4. Systemy Unix, Unix-based, Unix-like
5. Ideologia Unix

Zadania i komponenty 1

- Jądro (*Kernel*):
  - zarządza zasobami sprzętowymi komputera
  - zapewnia środowisko do uruchamiania procesów: zarządza ich uruchomieniem, dostępem do współdzielonych zasobów (mechanizm zamka [*lock*]), zapewnia interfejs do komunikacji pomiędzy procesami,
  - zarządza systemem plików
- Procesy komunikują się z jądrem systemu za pomocą zestawu tzw. *sys-call*i (zapytań systemowych),
- Interfejsem dla użytkownika jest powłoka (*Shell*) oraz zestaw programów, które implementują zapytania systemowe. Aplikacje użytkownika są odpalane z poziomu powłoki.

Zadania i komponenty 1

- Jądro (*Kernel*):
  - zarządza zasobami sprzętowymi komputera
  - zapewnia środowisko do uruchamiania procesów: zarządza ich uruchomieniem, dostępem do współdzielonych zasobów (mechanizm zamka [*lock*]), zapewnia interfejs do komunikacji pomiędzy procesami,
  - zarządza systemem plików
- Procesy komunikują się z jądrem systemu za pomocą zestawu tzw. *sys-call*i (zapytań systemowych),
- Interfejsem dla użytkownika jest powłoka (*Shell*) oraz zestaw programów, które implementują zapytania systemowe. Aplikacje użytkownika są odpalane z poziomu powłoki.

Zadania i komponenty 1

- Jądro (*Kernel*):
  - zarządza zasobami sprzętowymi komputera
  - zapewnia środowisko do uruchamiania procesów: zarządza ich uruchomieniem, dostępem do współdzielonych zasobów (mechanizm zamka [*lock*]), zapewnia interfejs do komunikacji pomiędzy procesami,
  - zarządza systemem plików
- Procesy komunikują się z jądrem systemu za pomocą zestawu tzw. *sys-call*i (zapytań systemowych),
- Interfejsem dla użytkownika jest powłoka (*Shell*) oraz zestaw programów, które implementują zapytania systemowe. Aplikacje użytkownika są odpalane z poziomu powłoki.

Zadania i komponenty 2

- Interfejsem do zapytań systemowych używanym przez binarne aplikacje są tzw. biblioteki (*libraries*) – SO (Unix) oraz DLL (Windows).
- Zalety:
  - Współdzielony kod dla wielu aplikacji – mniejsze obciążenie dla zasobów pamięciowych
  - Możliwość szybkiej aktualizacji błędów związanych z naruszeniem bezpieczeństwa systemu – aktualizacja jednej biblioteki załatwia sprawę wszystkich aplikacji, które się na niej opierały
- Wady:
  - Niebezpieczeństwo wstrzyknięcia złośliwej biblioteki, podszywającej się pod zaakceptowaną bibliotekę systemową
  - Konieczność utrzymywania bibliotek w wielu wersjach, dla wstecznej kompatybilności z niektórymi aplikacjami. Brak aktywnego zarządzania tym elementem systemu może prowadzić do zalegania ogromnej ilości bibliotek, które nie są używane przez ani jedną aplikację (zwłaszcza, choć nie tylko, z powodu wersjonowania).

Zadania i komponenty 2

- Interfejsem do zapytań systemowych używanym przez binarne aplikacje są tzw. biblioteki (*libraries*) – SO (Unix) oraz DLL (Windows).
- Zalety:
  - Współdzielony kod dla wielu aplikacji – mniejsze obciążenie dla zasobów pamięciowych
  - Możliwość szybkiej aktualizacji błędów związanych z naruszeniem bezpieczeństwa systemu – aktualizacja jednej biblioteki załatwia sprawę wszystkich aplikacji, które się na niej opierały
- Wady:
  - Niebezpieczeństwo wstrzyknięcia złośliwej biblioteki, podszywającej się pod zaakceptowaną bibliotekę systemową
  - Konieczność utrzymywania bibliotek w wielu wersjach, dla wstecznej kompatybilności z niektórymi aplikacjami. Brak aktywnego zarządzania tym elementem systemu może prowadzić do zalegania ogromnej ilości bibliotek, które nie są używane przez ani jedną aplikację (zwłaszcza, choć nie tylko, z powodu wersjonowania).

Zadania i komponenty 2

- Interfejsem do zapytań systemowych używanym przez binarne aplikacje są tzw. biblioteki (*libraries*) – SO (Unix) oraz DLL (Windows).
- Zalety:
  - Współdzielony kod dla wielu aplikacji – mniejsze obciążenie dla zasobów pamięciowych
  - Możliwość szybkiej aktualizacji błędów związanych z naruszeniem bezpieczeństwa systemu – aktualizacja jednej biblioteki załatwia sprawę wszystkich aplikacji, które się na niej opierały
- Wady:
  - Niebezpieczeństwo wstrzyknięcia złośliwej biblioteki, podszywającej się pod zaakceptowaną bibliotekę systemową
  - Konieczność utrzymywania bibliotek w wielu wersjach, dla wstecznej kompatybilności z niektórymi aplikacjami. Brak aktywnego zarządzania tym elementem systemu może prowadzić do zalegania ogromnej ilości bibliotek, które nie są używane przez ani jedną aplikację (zwłaszcza, choć nie tylko, z powodu wersjonowania).

Plan zajęć

1. FLOSS.
2. Zadania i komponenty systemu operacyjnego.
3. POSIX, SUS a Unix
4. Systemy Unix, Unix-based, Unix-like
5. Ideologia Unix

Portable Operating System Interface for uniX

- POSIX to zestaw standardów IEEE 1003, które określają zestaw niezbędnych zapytań systemowych, a także syntax powłoki i podstawowych programów pomocniczych.
- Systemy spełniające POSIX są identyczne z perspektywy procesowania kodu aplikacji użytkownika, różnice mogą kryć się jedynie w implementacji jądra/zapytań systemowych.
- Aplikacje odwołujące się jedynie do POSIX-owych zapytań systemowych będą działać jednakowo na wszystkich systemach spełniających POSIX

Single Unix Specification

- Standard SUS zotał utworzony jako wyraźnie lżejsza alternatywa dla standardu POSIX.
- Od 1997, odbywają się wspólne aktualizacje standardów POSIX oraz SUS pod kuratelą Open Group oraz IEEE (*Austin Common Standards Revision Group*).
- Ostatnia wydanie to SUSv4 (2008), znane również pod nazwą POSIX:2008 (a formalnie: IEEE Std 1003.1-2008). Poprzednie wydanie jest powszechnie nazywane SUSv3 albo POSIX:2001 (formalnie: IEEE Std 1003.1-2001).

UNIX

- Przez wiele lat *UNIX* było po prostu nazwą konkretnego, własnościowego, systemu operacyjnego.
- Z dzisiejszej perspektywy, każdy system spełniający standard *SUS* może być nazywany Uniksem.
- Systemy komercyjne, jak AIX, HP-UX, IRIX, Solaris, MacOS, Tru64, są *Uniks*ami.
- Większość systemów FLOSS, z Linux/GNU oraz *BSD na czele, są w dużej mierze zgodne z SUS. Te nazywa się *Unix-like*.
- Większość informacji tego przedmiotu będzie dotykać aspektów dostępnych we wszystkich z tych systemów operacyjnych.

Plan zajęć

1. FLOSS.
2. Zadania i komponenty systemu operacyjnego.
3. POSIX, SUS a Unix
4. Systemy Unix, Unix-based, Unix-like
5. Ideologia Unix

Początki historii Uniksa i języka C

- 1969: pierwsza wersja systemu UNIX stworzona w AT&T Bell Labs.
- 1969-1972: Dennis Ritchie (Bell Labs) rozwija język programowania C, mając na celu stworzenie języka pozwalającego na pisanie programów niezależnych od systemu operacyjnego.
- 1972: AT&T Unix przepisany w języku C.
- Z tego względu to C (lecz nie C++) jest najbardziej naturalnym językiem do programowania w Uniksie.
- Zapytania systemowe Uniksa są zaimplementowane jako funkcje w C.

AT&T Unix oraz Berkeley Unix

- Prawo przeciwmonopolowe z 1956 nie pozwala Bell Labs na komercyjne wejście na rynek komputerowy.
- Z tego względu AT&T decyduje się upowszechniać Unix bezpłatnie na uniwersytetach oraz instytutów naukowych.
- Grupa badawcza na Uniwerystecie Kalifornijskim w Berkeley, w ramach grantu DARPA, bierze Unix na warsztat, przemodelowuje go i integruje doń protokół TCP/IP.
- Dalsze wydania Unixa z Berkeley obejmują *1.xBSD* (1978), *2.xBSD* (1979), *3.xBSD* (1979) i *4.xBSD* (1980).
- Wzajemna stymulacja działa także w drugą stronę: AT&T wypuszcza *System III* (1982) oraz *System V* (1983) w oparciu o koncepcje zaczerpnięte z systemów BSD.

Komercyjne SystemV

- 1983: Departament sprawiedliwości rozstrzyga pozew antymonopolowy przeciw AT&T Bell Labs, w wyniku którego Bell Systems jest podzielone na mniejsze spółki, które już nie podlegają prawu antymonopolowemu. AT&T wycofuje Unix (SystemV, w skrócie SysV) z otwartego obiegu i zmienia w zamknięte, własnościowe oprogramowanie.
- Jednocześnie zamyka kanały współpracy z uczelniami i instytutami badawczymi.
- Licencja na kod SysV została zakupiona przez kilka innych firm, które dalej system rozwijały. Tak powstały m.in. *AIX* (IBM), *IRIX* (Silicon Graphics), *HP-UX* (Hewlett-Packard), *Solaris* (SUN).

Systemy *BSD

- W wyniku spraw cywilnych, BSD w formie czerpiącej z kodu AT&T Unixa przestał być uznawany za legalny. We wczesnych latac 90 powstaje *4.4BSD-Lite*. To otwarty (o licencji BSD) system Uniksowy bazujący 4.4BSD, lecz z całym kodem uprzednio wziętym żywcem z AT&T Unix – napisanym od nowa.
- Wydanie w 1995 *4.4BSD-Lite2* kończy historię Uniksa pisanego na Uniwersytecie Kalifornijskim w Berkeley.
- Kod 4.4BSD-Lite był później kontynuowany i rozwijany w społecznościowych systemach operacyjnych jak *NetBSD*, *FreeBSD* czy *OpenBSD*. O tych systemach mówi się, że należą do rodziny *BSD.

Projekt GNU

- 1983: Richard Stallman uruchamia *GNU Project*, którego celem jest stworzenie kompletnego systemu zgodnego z Uniksem, lecz otwartego i wolnego.
- GNU oznacza *GNU's not Unix!*
- 1983-1985: R. Stallman odchodzi z MIT na emeryturę i w pełni angażuje się w projekt GNU – zakłada *Free Software Foundation*, która kieruje, sponsoruje i promuje projekt GNU.
- 1989 pierwsza wersja licencji GNU GPL wychodzi na światło dzienne.

GNU/Linux

- We wczesnych latach '90 gotowy jest system operacyjny GNU, oraz istnieje prototyp jego jądra (*Hurd* – serce systemu).
- 1991: Linus Torvalds udostępnia swoje jądro *Linux*, kompatybilne z systemem GNU oraz licencją GNU GPL.
- Rodzina GNU/Linux to rodzina wolnych systemów **-nix*owych, opartych o jądro Linuxa oraz oprogramowanie GNU.
- Wśród nich najpopularniejsze to: *Arch*, *Debian*, *Fedora*, *Gentoo*, *Madriva*, *SuSE*, *Ubuntu*.

Systemy *-nixowe

Wśród systemów wywodzących się z oryginalnego systemu UNIX (w skrócie: systemów *-nixowych), wyróżnić można:

- własnościowe Uniksy, oparte o zakup licencji na kod AT&T SystemV, wzbogacony o część rozszerzeń systemów BSD. Wśród nich: *AIX*, *HP-UX*, *IRIX*, *Solaris*;
- własniściowe systemy oparte o głównie o kod BSD, wśród nich m.in. *MacOS, Tru64*;
- wolne/otwarte systemy operacyjne wywodzące się z BSD (4.4BSD-Lite – bez kodu własnościowego AT&T), m.in. *FreeBSD, OpenBSD, NetBSD*;
- wolne, otwarte, licencjonowane przez GNU GPL, napisane od zera, systemy GNU/Linux.

Plan zajęć

1. FLOSS.
2. Zadania i komponenty systemu operacyjnego.
3. POSIX, SUS a Unix
4. Systemy Unix, Unix-based, Unix-like
5. Ideologia Unix

Poniższe zasady podsumowują ideologię systemów UNIX:

- Rozdzielenie interfejsu użytkownika od procesowania
- Zarządzalność tekstowa
- Jawność/bezpośredniość
- Modularność
- Interfejs plikowy
- Rozdział uprawnień
- *Keep it Simple, Stupid!*

Zarządzalność tekstowa

- łatwość w edycji, zarządzaniu (przez człowieka) plików ASCII
- Komendy również są literałami
- Pliki konfiguracyjne rezydują w */etc* – (Editable Text Configuration / Edit-To-Configure)
- Skrypty Shellowe są plikami tekstowymi
- Logowanie odbywa się przez pisanie na ekran/do pliku

Interfejsy plikowe

- Dostęp do plików jest strumieniowy
- Sprzęt (m.in. terminal), pseudo-sprzęt widziany jako pewne specjalne pliki.
- Jednolity mechanizm czytania/zapisu do strumienia, niezależnie, czy powiązany on jest z plikiem, innym procesem, sprzętem itd.

Zarządzalność tekstowa

- łatwość w edycji, zarządzaniu (przez człowieka) plików ASCII
- Komendy również są literałami
- Pliki konfiguracyjne rezydują w */etc* – (Editable Text Configuration / Edit-To-Configure)
- Skrypty Shellowe są plikami tekstowymi
- Logowanie odbywa się przez pisanie na ekran/do pliku

Interfejsy plikowe

- Dostęp do plików jest strumieniowy
- Sprzęt (m.in. terminal), pseudo-sprzęt widziany jako pewne specjalne pliki.
- Jednolity mechanizm czytania/zapisu do strumienia, niezależnie, czy powiązany on jest z plikiem, innym procesem, sprzętem itd.

Modularność

- Programy/moduły mają mieć mały, ograniczony zakres odpowiedzialności, dzięki czemu łatwo je utrzymać. Bardziej skomplikowane systemy są z nich budowane jak z klocków
- Wielowarstwowa architektura pozwala na łatwą wymianę jednego komponentu na inny, równoważny (w sensie interfejsu, tj. postaci inputu i outputu).

Jawność

- Pełna kontrola nad kaskadą zapytań systemowych.
- Instalacja i konfiguracja może odbywać się w pełni manualnie (cf. zarządzalność tekstowa)
- *Klient ma zawsze rację.*

Modularność

- Programy/moduły mają mieć mały, ograniczony zakres odpowiedzialności, dzięki czemu łatwo je utrzymać. Bardziej skomplikowane systemy są z nich budowane jak z klocków
- Wielowarstwowa architektura pozwala na łatwą wymianę jednego komponentu na inny, równoważny (w sensie interfejsu, tj. postaci inputu i outputu).

Jawność

- Pełna kontrola nad kaskadą zapytań systemowych.
- Instalacja i konfiguracja może odbywać się w pełni manualnie (cf. zarządzalność tekstowa)
- *Klient ma zawsze rację*.

Rozdział uprawnień

- Konto administratora (*root*) ma nieograniczony dostęp do wszystkiego w systemie.
- Każdy proces uruchamiany jest z uprawnieniami użytkownika, który go odpalił.
- Procesom zwykle wystarczy znacznie mniej uprawnień, niż mają regularni użytkownicy (albo *root*).
- Z tego względu, zarządza się bezpieczeństwem przez tworzenie sztucznych użytkowników, którzy stają się właścicielami procesów. Taki użytkownik nie może się zalogować do systemu ani wystartować terminala.
- Dla łatwości zarządzania uprawnieniami, pliki różnych typów (pliki wykonywalne, biblioteki, pliki konfiguracyjne, bazy danych, logi etc) są położone w różnych lokalizacjach systemu plików.

Systemd

- Przeczy ideologii Uniksa.
- Super-moduł biorący na siebie odpowiedzialności wielu małych modułów, dla zoptymalizowania wewnętrznej komunikacji.
- Niekompatybilny z systemami Unix, zaprojektowany specjalnie dla Linuksa.
- Wzorowany na Windowsowym rozwiązaniu *svchost.exe*
- Trudny w konfiguracji i zarządzaniu (zapojmijcie o interfejsie plikowym!), błędy są praktycznie nienaprawialne/nieobchadzalne.
- Zwiększa wydajność w zwykłych, desktopowych, zastosowaniach