Podstawowy kurs Linuksa

Łukasz Koszyk

Gigaset, Wrocław, Kompilacja: 9 czerwca 2017, 11:37

Spis treści

Część I
Sylabus

1. Unix - podstawowe reguły
2. Unix - systemy pochodne
3. Linux - Kernel - dystrybucja
4. Struktura katalogów
5. Konsola a środowiska graficzne
6. Coreutils
7. Poruszanie się po katalogach
8. Użytkownicy i grupy
9. Prawa dostępu
10. System pomocy linuksowej
11. Podstawowe komendy
12. Potoki, strumienie, przekierowania
13. Skrypty
14. Przetwarzanie strumieniowe tekstu
15. Edytory tekstu
16. Praca z plikami - wyszukiwanie, podstawianie itp.
17. Urządzenia
18. Historia w Bashu
19. Systemy pakietowe
20. Procesy
21. Usługi
22. Start systemu - montowanie dysków, uruchamianie usług
23. Zdalna konsola
24. Cron crontab
25. Sieci w Linuksie
26. Debugowanie sieci, przechwytywanie pakietów, odpytywanie DNSów, listowanie połączeń, firewall

Część II
Wprowadzenie

Niniejszy dokument stanowi krótkie wprowadzenie do systemów Linux, w podstawowym zakresie. W swoim założeniu ma służyć jako uzupełnienie kilkudniowego kursu praktycznego obsługi Linuksa, natomiast przy odrobinie determinacji można z jego pomocą zapoznać się z omawianymi systemami we własnym zakresie.

Wiele zagadnień zostało uproszczonych, żeby uchwycić istotę rzeczy, a nie skupiać się na detalach i niuansach. Nie należy tego podręcznika traktować jako wyczerpującego omówienia systemów Linux.

Podczas nauki pomocny może tu być też taki skrypt: `http://www.math.uni.wroc.pl/~gosia/zajecia/skrypt.pdf`

Część III
Kilka słów o Linuksie

1. Linux czy Linuks, czy może linuks?

Spolszczanie nazw to temat rzeka. Jedni zalecają lecieć po całości, inni powstrzymywać się od tego procederu w przypadku nazw własnych. Pewną wykładnią może być wypowiedź prof. Bańki: `http://sjp.pwn.pl/slowniki/Linux.html`
Obecna wersja słownika podaje pisownię zaczynającą się od dużej litery z „x"w mianowniku, oraz spolszczonymi końcówkami w pozostałych przypadkach: Linux (inform.) -uksa a. -uxa, -uksie a. -uxie. Należy pamiętać, że jest to pisownia wzorcowa. Potoczna jest bardziej liberalna.

2. Unix

Nie wdając się zbytnio w historię - sterowanie pierwszymi komputerami elektrycznymi zbudowanymi na lampach elektronowych polegało na ręcznym ustawianiu parametrów (np. do obliczania równań różniczkowych) na warstwie elektrycznej poprzez przełączanie styków elektrycznych (czyli podchodził pan i przekładał kabelki i kręcił pokrętłem). Potem wprowadzono opcję sterowania za pomocą nośników danych - kart i taśm perforowanych. W międzyczasie zaczęto instalować pamięć operacyjną (rtęciową, magnetostrykcyjną, bębnową, ferrytową, półprzewodnikową), lampy zmieniono na tranzystory, wprowadzono pamięć masową (bębnową, magnetyczną na taśmach, magnetyczną dyskową), zaś sam komputer zaczął pracować pod kontrolą programu: systemu operacyjnego. Pierwsze generacje komputerów miały ściśle specjalizowane oprogramowanie, którego obsługa i utrzymanie wymagało dużej specjalistycznej wiedzy od inżyniera, także technicznej. Zaczęto więc pracować nad systemami operacyjnymi, które byłyby przenośne (przy okazji pracowano nad odpowiednimi językami programowania) zaś same komputery udoskonalano tak, aby dało się nimi sterować samym oprogramowaniem. I z tej idei wyszedł system UNIX zaprojektowany przez firmę AT&T, właściciela firmy Bell. Z oryginalnego UNIXa wyszedł m.in. system BSD (i jego odmiany Free-BSD, NetBSD itd) - na uniwersytecie Berkley zainstalowano system na komputerach PDP, a następnie zaczęto go udoskonalać na własne potrzeby. Naukowcy odsyłali poprawki do firmy Bell, wypuszczając przy okazji kolejne wersje systemu Berkeley Software Distribution. Pełna lista pochodnych gałęzi tutaj: `https://upload.wikimedia.org/wikipedia/commons/thumb/7/77/Unix_history-simple.svg/1920px-Unix_history-simple.svg.png`, a niezwykle złożona historia Unixa tutaj: `https://en.wikipedia.org/wiki/History_of_Unix`

Na tym etapie zatrzymamy się na chwilę. Systemy uniksowe posiadają pewne swoje założenia i paradygmaty. Z pomocą przychodzi tutaj całkiem niezłe opracowanie z Wikipedii: `https://en.wikipedia.org/wiki/Unix_philosophy`. Wedle tych paradygmatów oprogramowanie ma być proste, czytelne, robić jedną rzecz, ale dorze - reguła KISS. Ma także przedkładać przenośność nad efektywność wykorzystania zasobów sprzętowych. Wejście i wyjście programów ma być przygotowane do pracy w potokach. Konfiguracja ma być jawna, w plikach, opisana w czytelny dla człowieka sposób. Zaś sam system plików ma być hierarchiczny, każdy plik ma się znajdować w odpowiednim miejscu, zgodnie z jego przeznaczeniem. System operacyjny zapewnia wieloprocesowość, jawność i transparentność uruchomionych procesów i wszystkich pozostałych składników systemu. Konsekwencją tego jest fakt, że domyślnie użytkownicy mogą przeglądać swoje pliki, mogą przeglądać większość plików systemowych itd. Czyli podejście jest znacząco odmienne niż w przypadku systemow Microsoftu. Po co ten dziwaczny wstęp i baśnie o starych komputerach? Jak się za chwilkę czytelnik przekona - tak wygląda Linux i idea otwartego oprogramowania obecnie.

3. Linux

Gdzie w tym wszystkim jest Linux? Ano Linux jest systemem pochodnym, nazywanym uniksopodobnym. Dziedziczy po tym systemie wiele cech (część odrobinę liberalizując - dopuszczone jest opro-

gramowanie złożone, pisane w innych językach niż C, większy nacisk kładziony jest na przyjazność). W kolejnych częściach omówione zostaną podstawowe aspekty budowy systemu.

4. Linux - Kernel - dystrybucja

- Linux - system operacyjny oparty na jądrze Linux
- Kernel - samo słowo oznacza jądro systemu operacyjnego, choć najczęściej jest to skrót myślowy - Kernel (pisany zwłaszcza dużą literą) oznacza TEN kernel (jądro Linuksa)
- Dystrybucja - Kernel + oprogramowanie towarzyszące = gotowy do użytku system operacyjny. Jak makaron - mąka, woda, sól są te same, ale finalny produkt jest w różnych kształtach, z różnymi dodatkami i nadaje się do różnych portaw.

Kernel linuksowy jest aktywnie rozwijany i jego kod można pobrać tutaj: `https://www.kernel.org/`. Jest to tzw. vanilla kernel. Dlaczego z przymiotnikiem? Ano dlatego, że kernele dostępne w różnych dystrybucjach są zmodyfikowane przez deweloperów danej dystrybucji (są to mniejsze lub większe zmiany). Tutaj kilka słów o vanilla software: `https://en.wikipedia.org/wiki/Vanilla_software`.

W związku z tym nasuwa się pytanie: czy nie może być jednego Linuksa, tak jak jest jeden Windows (na dodatek z ładnymi numerkami, gdzie wiadomo, że 10 jest nowsze niż 7 czy 8)? Odpowiedź jest prosta: nie. Kernel jako projekt otwartoźródłowy może być użytywany zgodnie z licencją w różnych projektach. Samo to powoduje już, że nigdy nie będzie istniał jeden Linux. Stąd w niemal każdej dystrybucji, która również musi być projektem o otwartym źródle, jest kernel z łatkami, ale dla zachowania porządku stosuje się numerację oryginału z dopiskiem wskazującym na numer poprawki. Co ciekawe Microsoft użył kodu *BSD (czyli Unixa) w swoim systemie (systemy *BSD są dostępne na licencji BSD [sic!] która jest baaardzo liberalną licencją, i pozwala na sprzedawanie kodu w oprogramowaniu zamkniętym). Mało tego, MS ostatnimi czasy adaptuje narzędzia Unixowe w swoich systemach: `https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/`. Tak, ten sam Microsoft, który uważał kiedyś Linuksa i otwarte oprogramowanie za największą zarazę, i zwalczał te projekty wszelkimi możliwymi sposobami.

5. To ile tych linuksów i unixów jest?

Dużo. Tutaj stronka: `http://distrowatch.com/`

6. Po co tego jest aż tyle?

Odpowiedź brzmi: bo można :) Każda dystrybucja ma swoje przeznaczenie i powstała bo:

- ktoś chciał sobie podłubać
- potrzeba było obsłużyć konkretny sprzęt
- dostarcza jakichś unikalnych lub odmiennych od oryginału właściwości (np. inne niż oficjalne środowisko graficzne, kompilacja oprogramowania, które jest optymalizowane np. do przetwarzania dźwięku i zapewnia bardzo krótkie czasy)
- deweloperzy się pokłócili
- pojawiło się nowe zapotrzebowanie na rynku (np. urządzenia dotykowe, praca w chmurze)
- zainspirowano się oprogramowaniem zamkniętym (często absurdalnie drogim)

- korporacja stojąca za projektem (jej pracownicy angażują się w rozwój oprogramowania, ale też może to robić ktokolwiek spoza - tzw. społeczność) „poleciała w kulki"i ludzie ze społeczności spakowali manele, skopiowali kod, i poszli sami go rozwijać (np. OpenOffice → LibreOffice)

Stąd też się wziął sam Linux - Linus Torvalds hobbystycznie zaczął tworzyć wolny (w sensie wolności, nie spowolnienia) system operacyjny na ówczesne procesory Intela: 80386 oraz 80486 (znane jako i386, i486DX, i486SX).

7. Czołowe dystrybucje

Jest kilka czołowych dystrybucji głównych (Debian, RedHat, Slackware, Arch, Android), od których z czasem powstawały dystrybucje pochodne. Poniżej mocno skrócona lista głównych dystrybucji i ich pochodnych tworów opatrzona lakonicznym powodem powstania.

- Debian:
  - Ubuntu (prostszy, częściej wydawany):
    - Kubuntu (Ubuntu ze środowiskiem graficznym KDE)
    - Xubuntu (Ubuntu ze środowiskiem graficznym XFCE)
    - Lubuntu (Ubuntu ze środowiskiem graficznym LXDE)
  - Knoppix (System uruchamiany bezpośrednio z płyty CD)
  - Raspbian (Debian dla RaspberryPI)
- Slackware:
  - Slax
  - SuSE:
    - OpenSUSE (Darmowa wersja SuSE Enterprise)

  RedHat/RHEL:
  - Fedora (Najświeższe oprogramowanie, częste wydania, testowanie technologii na potrzeby RHEL):
    - Pidora (Fedora dla RaspberryPI)
    - Fedberry (Fedora dla RaspberryPI)
  - CentOS (Darmowa wersja RHEL)
  - Oracle Linux
  - Mandrake/Mandriva/OpenMandriva
- Gentoo
- Arch
- Puppy (Niewielka dystrybucja LIVE na słabe maszyny)
- Android (Smartfony, rozwijany obecnie przez Google):
  - Android-x86 (Instalacja androida na procesorach zgodnych z x86)
  - CyanogenMod (zwiększa funkcjonalność smartfona, przyspiesza, dostarcza aktualizacji):
    - LineageOS (Kontynuacja zamkniętego projektu CyanogenModa)

tutaj próba uszeregowania dystrybucji na obrazku: `https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg`. Jest tego trochę...

8. Jakie distro do czego?

Porządkując cały bałagan, usystematyzujmy do czego jaki Linux jest używany:

- Debian - gałąź stable dobra na serwery, gałąź testing na desktop
- Ubuntu - desktop, rzadko serwery
- CentOS (darmowy klon RHEL) - serwery
- Fedora - desktop (zwłaszcza pod bardzo nowy sprzęt)
- „drobne dystrybucje"typu Puppy, Tiny - bezdyskowe stacje robocze, odpalane przez PXE, bardzo stare maszyny
- OpenWRT, DDWRT i klony - routery sprzętowe
- Android i LineageOS - smartfony
- Raspbian, Fedberry - dystrybucje specjalizowane pod RaspberryPI
- PC-BSD - klon FreeBSD (a więc UNIX) desktop
- FreeBSD, OpenBSD, NetBSD - (również UNIX) serwery

Ze względu na zaawansowanie użytkownika (przykłady dystrybucji):

- Nowicjusz: Ubuntu i inne *buntu, PC-BSD, Mint
- Minimalne doświadczenie: Fedora, Debian
- Serwery: Debian, CentOS, FreeBSD, OpenBSD, NetBSD
- Doświadczeni użytkownicy: Gentoo
- Systemy specjalizowane: Raspbian, Fedberry, Debian/ARM, Fedora/ARM, Android, LineageOS, *WRT

9. Architektury systemów

Każda dystrybucja jest tworzona z myślą o konkretnej architekturze sprzętu (choć wiele jest dostępnych na kilka-, kilkanaście architektur sprzętu) np.

- i386, i486, i686 - komputery z 32-bitowymi procesorami klasy Intel Pentium lub Intel 486DX
- x86_64, amd64 - komputery z 64-bitowymi procesorami klasy Intel Pentium
- arm, armhf, armhfp, armv7l - komputery z procesorami ARMv7
- aarch64, arm64 - komputery z procesorami ARMv8

Są jeszcze wersje pod sparc, powerpc, IBM S390 itd. Każdy znajdzie coś dla siebie.

Część IV
Konsola a środowiska graficzne

Dział jest zasadniczo o interfejsie użytkownika, jednak zanim przejdziemy do omówienia szczegółów, należy przyjrzeć się uwarunkowaniom historycznym. Cofając się do zamierzchłych czasów - pierwsze komputery ustawiało się na warstwie elektrycznej manipulując przewodami i potencjometrami. W miarę postępu techniki pojawiały się nie tylko coraz to poręczniejsze nośniki danych (karty i taśmy perforowane) ale i terminale robocze podłączone do jednego komputera w budynku: człowiek pisał na klawiaturze, a komputer wypisywał odpowiedzi na drukarce. Mamy na to nawet ładną polską nazwę: dalekopis, z angielskiego TTY - teletype lub teletypewriter (zapamiętaj ten skrót: TTY). Potem, wraz z rozwojem techniki, drukarkę zastąpiono monitorem kineskopowym. Dlaczego tak komplikowano architekturę? Komputery w erze lampowej były bardzo drogie i prądożerne. Do pracy wymagały też odpowiedniej temperatury i wilgotności powietrza. Dlatego aby wykorzystać w całości moc obliczeniową, podłączano do centralnego komputera wiele terminali.

Drobna dygresja: wówczas na uczelniach studenci uczyli się programowania... na kartce papieru. Pisano kod ręcznie lub na maszynie do pisania, następnie należało pójść na uczelnię i w okienku oddać pani kod do przepisania. Po kilku dniach dostawało się kartę perforowaną z kodem programu i wydruk z komputera przedstawiający wyjście programu.

Wracając do tematu - upowszechnienie tranzystorów i spadek ich ceny doprowadził do powstania układów scalonych i pierwszych minikomputerów np. PDP-11 czy polskie MERA 300, MERA 400 (słowo mini należy rozumieć tutaj inaczej niż obecnie - w tamtych czasach mini było, gdy kilkadziesiąt szaf komputera zamknięto w jednej „gdańskiej trzydrzwiowej"), a przy dalszej miniaturyzacji do mikrokomputerów (wielkości standardowego PC albo Commodore C64). Mini i mikrokomputery obsługiwały w większości jeden terminal roboczy. Wtedy też po raz pierwszy człowiek miał do dyspozycji komputer osobisty.

Sprawa odwróciła się w momencie, gdy komputery zaczęto łączyć w sieci komputerowe. Wówczas praca w terminalach (tym razem wirtualnych) znów stała się popularna za pośrednictwem usług takich jak SSH, VNC, X-Window, RDP, Telnet. I tak jest do dziś...

Po omówieniu w kolejnych sekcjach w tej części wszystko stanie się jasne i zepnie się w jedną całość. Tytułem wstępu należy dodać, że obecnie użytkownik może pracować w dwóch trybach: tekstowym i graficznym.

10. Tryb tekstowy, konsola

Najprostszy sposób interakcji z systemem - zgodnie z intuicyjnym rozumieniem - wydaje się tekstowe polecenia komputerowi.

11. Tryb graficzny

Tu również dzisiejsze intuicyjne rozumienie jest dobre - użytkownik komunikuje się z komputerem generując zdarzenia w graficznym interfejsie.

12. Terminal tekstowy w Linuksie - zdalny i lokalny

W systemie Linuks/Unix można pracować zarówno zdalnie, jak i lokalnie. Lokalnie - logując się do systemu w trybie tekstowym (lub podpinając fizycznie pod JTAG). Wówczas użytkownik otwiera sesję TTY (znany skrót). Może się też połączyć zdalnie: kiedyś służył do tego protokół telnet (dzisiaj przestarzały), obecnie SSH, czyli Secured Shell. Połączenie jest bezpieczne, szyfrowane. Wówczas użytkownik otwiera sesję PTS (pseudo terminal slave). Najciekawsze jest to, że całość

jest przezroczysta dla oprogramowania, i w obydwu przypadkach działają tak samo. Zatem nie ma znaczenia, czy na Linuksie/Uniksie pracujemy zdalnie czy lokalnie. Efekt i praca jest taka sama (pomijam tu oczywiście specyfikę JTAG)

13. Środowiska graficzne w Linuksie - podstawa techniczna

Bez obaw, nie będę tu przepisywać dokumentacji, po co wprowadzono tryb graficzny - wiadomo. Jest to tak na prawdę program lub zespół współpracujących ze sobą programów. W Linuksie za tryb graficzny na ogół odpowiadają dwa komponenty: serwer wyświetlania oraz środowisko graficzne.

Serwerem wyświetlania jest X-Window System (oraz jego następca Wayland). Kwestia nazewnictwa - wszystkie X typu X-Window, X11, X11R6, Xwindows itp. to funkcjonalnie synonimy. Istnieją dwie implementacje: X.Org oraz XFree86. Komponent zajmuje się obsługą urządzeń wejściowych, rysuje okno, w którym program (lub środowisko graficzne) może umieścić obraz, oraz stanowi warstwę pośrednią dla środowiska graficznego. Tyle technikaliów na początek wystarczy

Środowisko graficzne - to tak na prawdę to, co widzi użytkownik. Wszystkie ikonki, okienka, paski itd.

14. Środowisko graficzne - praca lokalna, a zdalna

Praca lokalna w środowisku graficznym nie różni się od intuicyjnego rozumienia zagadnienia. Najciekawsza jest praca zdalna. Są dwa tryby pracy w takim środowisku.

Pierwszy - pulpit zdalny. Polega na tym, że lokalnie trzeba uruchomić serwer VNC. Lokalny serwer X-ów oraz środowisko graficzne tworzą obraz, który jest zrzucany do postaci pliku graficznego i cięty na kawałki. Fragmenty, które się zmieniły wysyłane są w czasie niemal rzeczywistym przez sieć. Akcje na kliencie są przechwytywane i wywoływane zdalnie, w szczególności ruchy myszki są widoczne. Z komputera może korzystać w trybie graficznym/przez VNC tylko jedna osoba.

Drugi - zdalne połączenie SSH z transportem X-ów. Wówczas obliczenia prowadzone są na zdalnej maszynie, a rysowaniem zajmuje się komputer klienta. Dlatego X-Window System nazywany jest serwerem wyświetlania, bo pracuje w klasycznej architekturze klient-serwer. Co ważne - jest to przezroczyste dla uruchamianego programu. Na jednej maszynie może być uruchomionych kilka takich sesji.

15. Wayland

Następca X-Window System. Obecnie (Maj 2017) jest oficjalnym serwerem wyświetlania jedynie w Fedorze 25. Planowane są migracje również w innych dystrybucjach Linuksa.

16. Wybór środowiska graficznego

Wyboru serwera wyświetlania dokonuje za nas zespół pracujący nad dystrybucją Linuksa, przygotowują łatki do jądra i kodu niektórych komponentów systemu. W sieci są dostępne poradniki jak użyć drugiego serwera.

Każda dystrybucja zazwyczaj ma też swoje domyślne środowisko graficzne. Ale można zainstalować sobie inne. Tutaj przykłady: `https://en.wikipedia.org/wiki/Desktop_environment`. Główna czwórka: KDE, GNOME, LXDE, Xfce. Środowiska różnią się między sobą efektami graficznymi, oprogramowaniem towarzyszącym, zasobożernością. Tutaj niedoskonałe porównanie: `https://en.wikipedia.org/wiki/Comparison_of_X_window_managers`.

Można też sobie zmienić ekran logowania: `https://en.wikipedia.org/wiki/X_display_manager_(program_type)#Some_implementations`. Ta lista również jest niepełna. Jak widać - jest w czym wybierać.

Część V
Obsługa konsoli linuksowej/uniksowej

Obsługa środowiska graficznego jest dość intuicyjna, jednak ma swoje smaczki, w zależności od tego jakie to środowisko. Z konsolą jest odwrotnie - trzeba znać kilka sztuczek, jest mniej intuicyjna na początku, ale jak się tego człowiek raz nauczy, to i na komputerze z lat 80 da sobie radę. Po uporaniu się z mentalnym strachem przed mityczną konsolą rodem z filmów amerykańskich (kiedy aktor wkłada pendrive na USB do stacji dyskietek i na ekranie same lecą literki) okazuje się, że często łatwiej coś zrobić w konsoli, niż klikać pół dnia.

17. Podstawy pracy z konsolą

Konsola jako narzędzie przysparza na początku wielu problemów i to normalne. Jest kilka podstawowych zasad, którymi należy się kierować przy jej używaniu.

17.1. Podstawowe polecenia

- exit
- groups
- man
- w
- whoami

17.2. Szukanie pomocy, co ja pacze?

Niemal każde polecenie ma swój system pomocy w konsoli. Na ogół pamięta się kilka opcji i kilkadziesiąt prostych najważniejszych poleceń. Pierwszym miejscem szukania pomocy jest użycie przełączników do polecenia `-h` lub `--help`. Następnie użycie polecenia `man <nazwa polecenia>`. Potem Google. **Nigdy nie należy kopiować poleceń na pałę do konsoli.** Należy zawsze sprawdzić każde polecenie i co robią jego przełączniki. Jeśli nie rozumiemy kodu, nie wpisujemy go do konsoli.

17.3. Przełączniki i opcje w poleceniach

Standardowo przełączniki jednoliterowe są po jednym minusie, a wieloliterowe po dwóch: `-h` i `--help`. Służą one do ustawiania parametrów poleceń.

17.4. Potoki

Polecenia w systemach uniksowych mogą pracować w potokach, to znaczy wyjście z jednego polecenia jest wejściem kolejnego. Potoki mogą mieć dowolną długość. Potok między dwoma poleceniami ustawia się kreską pionową: `|`. Przykład potoku: `w | wc -l` - zliczamy liczbę linii z wyjścia polecenia `w`.

17.5. Kod wyjścia

Każde polecenie zwraca kod wyjścia. Jeśli akcja kończy się sukcesem to jest to umownie 0, jeśli nie to 1. Maksymalnie można zwrócić liczbę 255. Odczytanie kodu wyjścia poprzedniego polecenia robimy najprościej poprzez `echo $?`

17.6. Katalog domowy

Każdy użytkownik ma swój katalog domowy, którego ścieżka zdefiniowana jest w pliku `/etc/passwd`. Jest to miejsce na wszystkie prywatne pliki tego użytkownika.

17.7. Plik i katalog

Katalog to obiekt struktury plików mogący przechowywać inne katalogi i pliki.

Plik to obiekt struktury plików, który w Linuksie występuje w pięciu typach:

- typowy plik
- dowiązanie symboliczne
- FIFO

- gniazdo
- urządzenie

Szczegóły tu: `https://en.wikipedia.org/wiki/Unix_file_types`

17.8. Wszystko jest plikiem

Jak widać wszystko w Linuksie jest plikiem - chcąc skomunikować się z procesem lub urządzeniem, po prostu pisze się do pliku. Naturalną konsekwencją hierarchicznej budowy systemu plików i powyższego paradygmatu, jest fakt, że narzędzia konsolowe mogą być na prawdę uniwersalne, w szczególności narzędzia do obróbki strumieniowej tekstu (omówione w osobnym rozdziale).

17.9. Standardowe strumienie i ich przekierowywanie

Są trzy standardowe strumienie: stdin, stdout, stderr. Każdy z nich ma przypisaną liczbę kolejno: 0, 1, 2. Wszystkie strumienie, i każdy z osobna można przekierowywać w różne miejsca (pliki). Pod tym linkiem kilka przykładów (strona ma braki): `https://pl.wikipedia.org/wiki/Standardowe_strumienie`, i trochę lepsza: `http://wiki.bash-hackers.org/syntax/redirection`

18. Katalogi, pliki, buszowanie po nich

Pierwsza ważna umiejętność - manipulacje plikami i katalogami.

18.1. Struktura katalogów

W systemach uniksowych mamy zasadniczo do czynienia z tą samą strukturą katalogów. Główna partycja jest montowana pod `/` i nazywamy ją rootem (slashem, rzadko korzeniem). Struktura katalogów jest ściśle hierarchiczna. Oto typowa zawartość głównego katalogu:

- /bin
- /boot
- /dev
- /etc
- /home
- /lib
- /lib64
- /media
- /mnt
- /opt
- /proc
- /root
- /run
- /sbin
- /srv
- /sys
- /tmp
- /usr
- /var

A skrótowy opis przeznaczenia każdego katalogu tutaj: `https://wiki.fedora.pl/wiki/Podstawy_Linuksa#Partycje_i_system_plik.C3.B3w`

18.2. Przydatne polecenia

- cat
- cd
- chmod
- chown
- cp
- df
- du

- find
- grep
- head
- ln
- ls
- man
- mkdir
- mv
- pwd
- rm
- sort
- tail
- touch
- uniq
- wc
- whoami

18.3. Ćwiczenia

1. Załóż katalog `/tmp/moj_katalog`. Wejdź do tego katalogu. Utwórz w nim pusty plik `m_plik`
2. Jednym poleceniem załóż katalog: `/tmp/a/b/c/d/e/f/g/h/i/j`
3. Wywołaj polecenie: `echo 'kuku' > /tmp/pliczek`
4. Używając polecenia `wc` policz liczbę linii i znaków w `/tmp/pliczek`
5. Wypisz na ekran zawartość pliku `/tmp/pliczek`
6. Skopiuj plik `/tmp/pliczek` do `/tmp/pliczek2`
7. Ustaw prawa odczytu i wykonywania dla siebie, prawa tylko do odczytu dla innych, na `/tmp/pliczek2`
8. Utwórz dowiązanie symboliczne z `/tmp/pliczek2` do `/tmp/pliczek_ln`
9. Przenieś `/tmp/pliczek_ln` do katalogu `/tmp/a/b/c/d/e/f/g/h/i/j`
10. Wejdź do katalogu `/tmp/a/b/c/d/e/f/g/h/i/j`, wykonaj polecenie `pwd > pliczek_ln`. Nie wychodząc z tego katalogu wypisz na ekran zawartość pliku `/tmp/pliczek2`
11. Znajdź w katalogu `/var` największy plik.
12. Który z podkatalogów katalogu `/var` zajmuje najwięcej miejsca na dysku?
13. Sprawdź ile masz wolnego miejsca na dysku (w megabajtach).
14. Znajdź w katalogu `/etc` wszystkie pliki z literką `c` lub `C` w nazwie.
15. Ile jest katalogów w całym `/sys` i jego podkatalogach. Ile tam jest plików?
16. Znajdź w katalogu `/etc` wszystkie pliki zawierające ciąg znaków: `mon`. Litery mogą mieć dowolną wielkość. Policz ile takich linijek znalazło.

Dodatkowe ćwiczenia można znaleźć tu: `http://www.math.uni.wroc.pl/~gosia/zajecia/lista1.pdf`

19. Użytkownicy i grupy

Tutaj całkiem niezły opis: `https://wiki.archlinux.org/index.php/users_and_groups`

19.1. Przydatne polecenia

- adduser
- chmod
- groupadd
- groupdel
- groups
- passwd
- userdel

20. Edytory tekstu

Jest wiele edytorów tekstu dostępnych w konsoli, do najpopularniejszych należą: vim, nano, pico, emacs, mcedit. Najpopularniejszym i najciekawszym jest vim. Nie ma sensu się rozpisywać - świetny kurs do nauki vima to `vimtutor`. Pomocny będzie 1,2,3 punkt na polskiej stronie Wikipedii: `https://pl.wikipedia.org/wiki/Vim`, opisane skrótowo, bez ględzenia.

20.1. Ćwiczenia

1. Przejść kurs vimtutor
2. Nauczyć się edycji blokowej
3. Nauczyć się sortować linijki
4. Nauczyć się kopiować i wklejać pojedyncze słowa/litery
5. Nauczyć się kopiować i wklejać kilka linii

21. Strumieniowa edycja tekstu

Użytkownicy programów graficznych przyzwyczajeni są do typowych graficznych edytorów. Vim, nano czy emacs chociaż w konsoli tekstowej, to są to edytory graficzne typu TUI. Można też używać edytorów wierszowych typu ed czy ex. Używano ich w czasach pierwszych komputerów, z drukarkami zamiast monitorów. Inne podejście prezentują edytory strumieniowe i narzędzia umożliwiające strumieniowe przetwarzanie tekstu. Dziedziczą one składnię po edytorach ed i ex. Po dziś dzień składnia ta jest używana również w innych edytorach typu TUI.

Edytorów strumieniowych używa się dziś równie chętnie, co TUI, ale do innych rzeczy. Stosuje się je gdy ilość tekstu jest bardzo duża, gdy trzeba na żywo odfiltrować wyjścia z poleceń, gdy na urządzeniu nie wystarcza pamięci operacyjnej do edycji w TUI lub GUI (bo plik za duży, albo urządzenie ma mało RAMu), gdy na przykład trzeba zmienić jedno słowo na inne w kilku tysiącach plików itd. Metoda polega na tym, że program aplikuje filtry linia po linii, może rozdzielić jedną linię na kilka zgodnie z filtrem, może jakieś usunąć lub zwielokrotnić - wiele poleceń łączy się w potok, z czego każde wykonuje jedną prostą czynność.

21.1. Przydatne polecenia

- awk
- cut
- grep
- head
- paste
- sed
- sort
- tail
- tr
- uniq

21.2. Informacje dodatkowe

- awk jest osobnym językiem interpretowanym, ale znakomicie nadaje się do edycji strumieniowej. Typowe zastosowanie to:
  - wypisanie konkretnych kolumn rozdzielonych jakimiś znakami: `cat plik | awk '{print $1" "$3"etrett"$7}'`
  - łatwe wypisanie ostatniej kolumny: `awk '{print $NF}'`
  - łatwo można ustawić „delimiter” - domyślnym jest spacja, grupa spacji, tabulator, grupa tabulatorów, ale można ustawić własny przełącznikiem -F, np: `grep -irs mon /etc | awk -F ':' '{print $1}'`
  - szybko można zsumować liczby: `awk '{sum += $3} END {print sum}'`
  - i inne ciekawe rzeczy: `https://pl.wikipedia.org/wiki/AWK`
- cut pozwala łatwo rozdzielić plik, czasem łatwiej niż awk: `https://pl.wikipedia.org/wiki/Cut_(Unix)`
- grep to podstawowe polecenie, warto je dobrze poznać

21.3. Ćwiczenia

1. Wyświetl wszystkie pliki z katalogu `/etc` które zawierają ciąg `mon`. Ile ich jest?
2. Wypisz na ekran plik `/etc/fstab` ze spacjami zamienionymi na nowe linie
3. Podaj listę 7 największych plików z katalogu `/etc` wraz z rozmiarami
4. Podaj listę 10 plików z `/etc` które mają najwięcej linii tekstu
5. Ile plików z katalogu `/usr` ma nazwy które się powtarzają?
6. Podaj listę rozszerzeń plików z katalogu `/usr` wraz ze statystyką użycia (wystarczy liczba wystąpień).

22. Inne przydatne polecenia

Poniżej strony z innymi przydatnymi poleceniami w konsoli:

1. `https://pl.wikipedia.org/wiki/GNU_Coreutils`
2. `https://en.wikipedia.org/wiki/List_of_Unix_commands`
3. `https://ss64.com/bash/`

23. Skrypty w Bashu i nie tylko

Jednolinijkowców używa się tylko do jakichś prostszych rzeczy, bardziej skomplikowane łatwiej zrobić za pomocą skryptów w Bashu. W zasadzie cała dotychczasowa wiedza się przyda, dojdą tylko instrukcje warunkowe, pętle i inne proste i przydatne rzeczy.

23.1. Składnia Basha

Bash jest powłoką systemową dostarczającą interpretowanego języka. Składnia z przykładami tu: `https://ss64.com/bash/syntax.html` oraz `http://www.learnshell.org/`

23.2. Łatwe przeszukiwanie historii Basha

Tutaj fajnie opisane: `https://www.digitalocean.com/community/tutorials/how-to-use-bash-history-commands-and-expansions-on-a-linux-vps#scrolling-through-bash-history`

23.3. Ćwiczenia

1. Napisz pętlę, która wywoła polecenie stat na wszystkich plikach z rozszerzeniem mp4 w danym katalogu. Zrób to samo bez pętli

2. Napisz jednolinijkowiec który pobierze listę wszystkich plików i katalogów z bieżącego katalogu, zamieni w locie w ich nazwach literę `a` na `z` (bez zmiany nazwy obiektu w systemie plików), następnie posortuje alfabetycznie i używając pętli while wywoła odpowiednie polecenie, którego efektem będzie:
   `plik 1: <nazwa pliku1>`
   `plik 2: <nazwa pliku2>`
   `plik 3: <nazwa pliku3>`
   (...)
   A następnie używając polecenia `column` wyrówna ładnie kolumnami wyjście.

3. Napisz skrypt, który jako pierwszy parametr przyjmuje listę słów oddzielonych przecinkami, a drugi parametr liczbowy (nazwijmy go `n`). Skrypt sortuje słowa alfabetycznie, a następnie wypisuje na ekran słowo numer `n`

4. Napisz skrypt, który wywoła polecenie: `grep mon /etc/fstab > /dev/null` i jeśli `mon` zostanie znaleziony, to wypisuje na ekran `TAK`, w przeciwnym przypadku dodaje: `# mon` jako drugą linijkę do `/etc/fstab`

5. Załóżmy że są 3 pliki. Zawartość `num1`: `7452`, zawartość `num2`: `liczba klientow: 753`, zawartość `num3`: `cos tam cos tam: 754`, przy czym liczby mogą się zmienić. Napisz skrypt który wyciągnie liczby z tych trzech plików i wypisze je na ekran, każda w osobnej linii.

6. Używając awk i potoku zsumuj liczby z wyjścia z poprzedniego zadania: `./skrypt | <sumowanie>`

24. Urządzenia w Linuksie

Jako że wszystko w Linuksie jest plikiem, urządzenia też mają swoją reprezentację plikową. Standardowo są one w `/dev`, ewentualnie do niektórych funkcji można dostać się przez `/sys`. Szczegóły tutaj: `https://en.wikipedia.org/wiki/Device_file`. Szczególnej uwadze polecam wirtualne urządzenia.

24.1. Przydatne polecenia

- dd
- fdisk
- losetup
- mount
- mkfs

24.2. Ćwiczenia

1. Pobierz najnowszy obraz raspbiana: `https://downloads.raspberrypi.org/raspbian_lite_latest`. Wypakuj archiwum. Podejrzyj fdiskiem strukturę obrazu.
2. Zamontuj główną partycję z obrazu, odczytaj plik `etc/fstab`
3. Wypisz na ekran pierwsze 500 bajtów ze swojego lokalnego dysku
4. Używając polecenia dd i źródła z `/dev/zero` utwórz plik `dysk_testowy` rozmiaru 2048MB
5. Załóż w pliku `dysk_testowy` trzy partycje i sformatuj je na podane systemy pliku: pierwszą FAT 512MB, drugą ext4 1024MB, na pozostałej części trzecią xfs
6. Zamontuj te partycje do osobnych katalogów
7. Na każdej partycji utwórz po jednym pliku z dowolną zawartością. Odmontuj partycje
8. Znajdź metodę tworzenia obrazu ISO płyty CD. Jeśli masz napęd optyczny w komputerze - utwórz taki obraz. Jeśli nie masz, pobierz ten: `http://distro.ibiblio.org/puppylinux/puppy-slacko-6.3.0/32/slacko-6.3.0.iso`
9. Zamontuj obraz iso, wylistuj jego pliki
10. Weź jakiś pendrive. Zapisz na nim obraz `https://download.fedoraproject.org/pub/fedora/linux/releases/25/Spins/x86_64/iso/Fedora-Xfce-Live-x86_64-25-1.3.iso`. Uruchom tę dystrybucję LIVE na swoim komputerze. Spróbuj w niej zamontować swój dysk twardy i podejrzeć pliki

25. Systemy pakietowe w Linuksach

Instalacja dodatkowego oprogramowania jest jedną z podstawowych czynności. Można to robić na trzy sposoby: kompilując źródła oprogramowania (niepolecane), używając prekompilowanych źródeł (popularne w systemach BSD) lub używając dedykowanych systemów pakietowych. W Windowsie jest instalator, w Linuksie jest pakiet. W Windowsie jeden instalator dostarcza pełny program, w Linuksie pakiet ma swoje zależności do bibliotek i szczególny nacisk kładziony jest na deduplikację. Niczego nie trzeba pobierać ręcznie z internetu, aktualizacje całego oprogramowania również można robić jednym kliknięciem.

Mamy dwa najpopularniejsze systemy pakietowe: deb i rpm. Deb są dla dystrybucji debianowych i pochodnych, rpm dla RedHatów i pochodnych. Do każdego z systemów pakietowych dostępne są łatwe w użyciu narzędzia służące do automatycznego pobierania pakietów, aktualizacji, rozwiązywania zależności, przeszukiwania itd.

25.1. yum/dnf

Podstawowe narzędzie w pochodnych RedHata. W systemach RedHat 6,7, Centos 6,7, Fedora do 21 włącznie jest yum, w Fedorach 22 i nowszych jest dnf - następca yuma. Obydwa narzędzia korzystają z tego samego backendu - z rpm. Tutaj przyjemnie opisane jak używać yuma: `https://fedora.pl/poradnik#I-1`. A tu jak dnfa: `https://wiki.fedora.pl/wiki/Poradnik_F22_(Twenty_Two)#DNF`.

25.2. apt/aptitude

W pochodnych Debiana jest apt (lub aptitude). Całkiem ładnie jest to opisane w dokumentacji do Ubuntu: `https://help.ubuntu.com/community/AptGet/Howto`

25.3. Inne dystrybucje

W innych dystrybucjach często są własne systemy pakietowe np. yast w SuSE. Wszystko jest ładnie opisane w internecie.

25.4. Graficzne instalatory

Środowiska graficzne na ogół mają program, w którym można jeszcze łatwiej instalować, wyszukiwać itd. Są to graficzne nakładki na yuma, apta itd. Wszystko również można znaleźć w poradnikach do poszczególnych dystrybucji.

26. Procesy i zarządzanie nimi

Co to jest proces - wiadomo. Tutaj garść dodatkowych skrótowych informacji na ten temat: https://en.wikipedia.org/wiki/Process_(computing). Zarządzanie procesami w Linuksach można robić na kilka sposobów. Dla porządku pominiemy tutaj na chwilkę usługi. Zajmiemy się chodzącymi procesami

26.1. Przydatne narzędzia

- htop
- kill
- killall
- lsof
- nice
- ps
- pstree
- strace
- top

26.2. Dodatkowe informacje

Nie ma sensu się rozpisywać na ten temat - jest to jeden z dokładniej omówionych aspektów używania systemów Linux. W skrócie: procesy można zabijać, zmieniać im priorytet dostępu do procesora i innych zasobów, usypiać, zatrzymywać itd. Tutaj wystarczająca wiedza na początek: https://pl.wikibooks.org/wiki/Linux/Procesy, polecam tylko ominąć initctl.

27. Usługi w Linuksach

W Linuksach są dość rozbudowane narzędzia do zarządzania usługami startującymi podczas uruchamiania systemu (nie należy tego mylić z programami uruchamianymi po zalogowaniu przez użytkownika). Są zasadniczo trzy narzędzia do tego służące. Dwa starsze, wychodzące z użycia to SysVinit i Upstart, oraz nowe: SystemD. Tutaj krótka dokumentacja do SysVinit: http://www.tldp.org/HOWTO/HighQuality-Apps-HOWTO/boot.html, a tutaj do SystemD: https://www.digitalocean.com/community/tutorials/systemd-essentials-working-with-services-units-and-the-journal i tu: https://wiki.archlinux.org/index.php/systemd.

27.1. Ćwiczenia

1. Napisz skrypt w Bashu składający się tylko z pustej nieskończonej pętli, która coś ewentualnie wypisuje na standardowe wyjście.
2. Napisz usługę do SystemD uruchamiającą Twój skrypt
3. Sprawdź czy działa
4. Używając programu strace podejrzyj co usługa robi. Podejrzyj też inne usługi.

28. Cron

W skrócie - narzędzie do cyklicznego uruchamiania skryptów w Linuksie, o konkretnej godzinie, w konkretny dzień tygodnia, miesiąca, roku. Proste i bardzo przydatne narzędzie. Szczegóły tu: https://en.wikipedia.org/wiki/Cron i tu: https://wiki.archlinux.org/index.php/cron

29. Cgrupy

Dział ten jest w zasadzie uzupełnieniem wcześniejszych informacji. Cgrupy służą do bardzo dokładnego przydzielania zasobów procesom. Można przypiąć proces/procesy do konkretnego rdzenia procesora, limitować pamięć itd. Na co dzień się tego nie używa, raczej do bardziej wyrafinowanych zastosowań, ale warto pamiętać, że coś takiego jest, bo cgrupy pozwalają robić cuda niewielkim kosztem. Limity można bardzo łatwo ustawiać w usługach systemd. Tutaj fajnie opisane: `https://wiki.archlinux.org/index.php/cgroups` oraz tutaj: `https://en.wikipedia.org/wiki/Cgroups`

30. Proces uruchamiania Linuksa

Tutaj dość zdawkowo opisane: `https://en.wikipedia.org/wiki/Linux_startup_process`. Należy to kojarzyć z hasłami kernel, runlevel, GRUB, LILO, dracut, initramfs, initrd. Dla chętnych można doczytać i się wielu ciekawych rzeczy dowiedzieć.

To, co nas będzie interesować to aspekty praktyczne: jakie usługi startują na początku i jak je dodać - omówione powyżej. Pozostała jeszcze jedna interesująca rzecz:

30.1. Montowanie partycji

W Linuksie głównym miejscem, gdzie się to ustawia jest `/etc/fstab`, i w tym miejscu należy montować swoje dodatkowe partycje (nie skryptami, ani własnymi usługami). Tutaj jest bardzo ładnie opisane co i jak: `https://wiki.archlinux.org/index.php/fstab`. Podczas startu systemu jest jeszcze montowane coś takiego, jak rootfs - informacja o tym znajduje się standardowo w GRUBie.

Czasem nie chcemy mieć na stałe zamontowanych partycji, albo np. dysków sieciowych (z różnych powodów). Wówczas można użyć narzędzia autofs. Wtedy zasób jest montowany przy dostępie do katalogu, gdzie powinien być zamontowany, a po odpowiednim okresie nieaktywności odmontowywany. Metoda ma tę zaletę, że przy problemach ze zdalnym dyskiem, system operacyjny bez problemu się uruchomi i zamknie. Gdybyśmy to montowali w fstabie, wówczas trzeba czekać kilka minut na timout. Tutaj więcej informacji na ten temat: `https://wiki.archlinux.org/index.php/Autofs`.

31. Konsola zdalna

Pozostał do omówienia jeszcze jeden temat - zdalny dostęp do maszyny. Standardowo służy do tego ssh. Zdalne kopiowanie plików przez scp. Zdalna synchronizacja katalogów - rsync+ssh.

31.1. Przydatne narzędzia

- ssh
- scp
- sftp
- rsync

31.2. Informacje

- `https://support.suso.com/supki/SSH_Tutorial_for_Linux`
- `https://www.digitalocean.com/community/tutorials/ssh-essentials-working-with-ssh-servers-clients-and-keys`
- `https://www.digitalocean.com/community/tutorials/how-to-use-ssh-to-connect-to-a-remote-server-in-ubuntu`

31.3. Sztuczki z SSH

SSH, oprócz tego że mamy zdalną konsolę, pozwala robić różne sztuczki:

- Tunelowanie Xów po SSH - po prostu `ssh -X`
- Tunelowanie ruchu na konkretne porty przez SSH - potężne narzędzie do debugowania i przekierowywania ruchu przez inne hosty: `http://www.revsys.com/writings/quicktips/ssh-tunnel.html` oraz tu: `http://blog.trackets.com/2014/05/17/ssh-tunnel-local-and-remote-port-forwarding-explained-with-examples.html`. Można tunelować jakikolwiek ruch idący po sieci
- Montowanie dysków przez SSH - tzw. sshfs. Powiedzmy, że chcemy mieć swój zdalny katalog domowy dostępny jakby był zamontowany lokalnie - `https://wiki.archlinux.org/index.php/SSHFS` oraz `https://www.digitalocean.com/community/tutorials/how-to-use-sshfs-to-mount-remote-file-systems-over-ssh`
- Synchronizacja zdalna katalogów - rsync + ssh
- Można używać potoków z SSH (np zdalna kompresja katalogu wysyłana przez ssh i rozpakowywana na miejscu)
- W szczególności można używać tego wtedy, gdy się chce ominąć proxy (filtrujące np. jakieś domeny) w korposieci
- W sumie - można użyć dowolnego polecenia i zrobić niemal wszystko, dokładając tylko jedną warstwę pośrednią, która jest całkowicie przezroczysta, np. można zdalnie zapisać obraz na dysk używając dd (w locie! bez uprzedniego przesyłania obrazu na maszynę), można odczytać dysk/zrobić ISO ze zdalnego napędu, również w locie.

Część VI
Sieci

32. Konfiguracja sieci

Konfiguracją sieci zarządza w Linuksach zazwyczaj usługa o nazwie NetworkManager, która ustawia sieć samoczynnie odpytując DHCPa. Całkiem wygodnie obsługuje się ją narzędziami GUI albo TUI. Przez CLI już trochę mniej. Szczegóły można znaleźć w poradnikach i dokumentacji do poszczególnych dystrybucji. Można też zrezygnować z NetworkManagera i sieć ustawiać ręcznie. Tutaj jak to zrobić w RedHatach i pochodnych: `https://www.howtoforge.com/linux-basics-set-a-static-ip-on-centos`, a tutaj w Debianie i pochodnych: `https://www.howtoforge.com/debian-static-ip-address`. Można też użyć polecenia dhclient.

33. Debugowanie sieci

Ważną umiejętnością jest debugowanie sieci, weryfikowanie tras itd. W Linuksie jest kilka narzędzi upraszczających tę sprawę.

33.1. Przydatne narzędzia

- ip
- ping
- telnet
- nc
- iperf
- tcpdump
- iftop

Narzędzie ifconfig jest już przestarzałe i nie należy go używać.

33.2. Niektóre zastosowania wymienionych narzędzi

- `ip r sh`, `ip r get`, `ip r add`, `ip r del` - tablice routingu: wyświetlanie, sprawdzanie którędy idą pakiety dla danego IP, dodawanie, usuwanie
- `ping`, `ping -6` - puszczanie pakietów protokołem ICMP; w nowszych dystrybucjach ping obsługuje IPv6 tak jak IPv4 - bez dodatkowych przełączników
- `telnet <host> <port>` - sprawdzanie czy jest łączność do usługi na porcie TCP; weryfikacja czy działa prawidłowo (wpisując odpowiednie komendy)
- `tcpdump -i any -nnnn`, `tcpdump -i any -nnnn port 80` - w obu przypadkach podglądanie ruchu na wszystkich interfejsach sieciowych, w drugim limitując do portu 80
- `tcpdump -i any -nnnn -vvvv -s0 not port 22 -w <plik>` - tcpdump tworzący plik, który można obejrzeć w wiresharku