SYSTEMY OPERACYJNE

Prowadzący: Antoni Ligęza
e-mail:aligeza@pwsz-ns.edu.pl
http://www.pwsz-ns.edu.pl/aligeza

Wszystkie brakujące rozwiązania zadań należy przesłać na podany powyżej adres mailowy, umieszczając je bezpośrednio w treści maila (nie załącznika). Temat maila ma być złożony z następującego tekstu: informatykaSO:NazwiskoStudenta:ImieStudenta. Termin nadsyłania rozwiązań upływa w dniu poprzedzającym następne zajęcia.

Spis treści

Część I

Laboratorium

ĆWICZENIE 1

Podstawowe pojęcia i narzędzia Linuxa

1.1. Krótki opis systemu Linux

1.1.1. Dystrybucje Linuxa

Dystrybucja - w odniesieniu do systemu operacyjnego Linux to zestaw programów rozpowszechnianych łącznie i dający po zainstalowaniu gotowy do użycia system.

Ponieważ pobranie i skompilowanie wszystkich programów składających się na system operacyjny wymagałoby od użytkownika dużo wiedzy, czasu i wysiłku, dosyć wcześnie (w historii systemów Linux) zaczęto tworzyć gotowe zestawy istotnie ułatwiające tę czynność.

We współczesnych dystrybucjach, programy występują zwykle w postaci pakietów (DEB, RPM, tarball). Dystrybucja oferuje system pobierania, instalacji, deinstalacji i uaktualniania pojedynczych pakietów (APT, DPKG, RPM), rozwiązujący zależności między pakietami oraz często wspólny interfejs konfiguracji pakietów (np. debconf).

Zazwyczaj dystrybucje są instalowane bezpośrednio na dysku stałym komputera. Istnieją również dystrybucje nie wymagające instalacji. Uruchamiane są one bezpośrednio z nośnika:

* CD-ROM, tzw. LiveCD, np. Knoppix, * pamięci USB, * (kiedyś prawie wyłącznie) dyskietki, np. Pocket Linux

lub z sieci przy użyciu NFS.

Większość dystrybucji jest rozpowszechniana na płytach CD-ROM i dostępna do pobrania przez Internet.

Z obecnie używanych dystrybucji Linuksa najdłuższą historię mają Slackware, Debian oraz Red Hat Enterprise Linux (który jest bezpośrednim rozwinięciem Red Hat Linux, zmiana nazwy miała charakter marketingowy). Na ich bazie powstało wiele innych samodzielnych dystrybucji.

Popularność dystrybucji bardzo się waha w zależności od kraju oraz języka użytkownika. W skali globalnej pewną miarą względnej popularności może być zestawienie tworzone przez DistroWatch.

1.2. Rozpoczęcie pracy

Konfiguracja VirtualBox, klonowanie obrazów, logowanie,

Uwaga Polecenia linuxowe rozróżniają małe i duże litery (polecenie dir jest innym poleceniem od Dir, DIR,dIR, itp., podobnie root jest innym użytkownikiem od Root)!!!

Zalogowanie na konto root.

Użytkownicy są identyfikowani w systemie nie tylko ma podstawie nazw. System przypisuje użytkownikom również identyfikatory numeryczne, które są wykorzystywane do identyfikacji prawa przynależności plików i procesów w systemie. Informacje o takim identyfikatorze uzyskaj wprowadzając polecenie id. Zwraca on również informacje o grupach.

1.2.1. mc - menedżer plików

podgląd F3 i edycja plików F4, zmiana katalogów, tworzenie F7, kopiowanie plików i katalogów

1.2.2. Konsole wirtualne

W Microsoft Windows można uaktywni? jednocześnie kilka okien, w których jednocześnie mogą być wykonywane różne czynności. Choć powłoka wiersza poleceń Linuxa nie ma graficznego interfejsu użytkownika, umożliwiającego jednoczesne uaktywnienie kilku takich okien to można jednak pracować z kilkoma egzemplarzami powłoki jednocześnie posługując się do tego celu **konsolami wirtualnymi**. Czyli nawet, jeśli do komputera podłączona jest tylko jedna fizyczna konsola (monitor + klawiatura) to możliwe jest jednoczesne logowanie na wiele różnych kont. Linux Peanut udostępnia 3 konsole wirtualne. Użycie kombinacji klawiszy [Alt+Fn], gdzie n jest numerem konsoli wirtualnej (1-2), spowoduje, ze Linux wyświetli konsole o numerze n. System X Windows działa na konsoli 3 (w dystrybucji RedHat jest nią siódma konsola). Aby z X Windows przejść do konsoli (1-2) używamy sekwencji klawiszy [Ctrl-Alt-Fn]. Konsoli wirtualnych można używać do jednoczesnego wykonywania kilku czynności, jak na przykład jedną konsolę można wykorzystać do pisania a na drugiej można zainstalować jakiś nowy pakiet oprogramowania zajmujący dużo czasu albo wyświetlić jedną z lekcji, które w przyszłości będą przekazywane w formie elektronicznej.

Zadanie 1.1. Przejść do konsoli wirtualnej nr 2 i zalogować się jako superużytkownik. Następnie poleceniem who można sprawdzić, kto jest aktualnie zalogowany w naszym systemie i na której konsoli (konsole oznaczane są przez ttyn) Poleceniem who am i można dowiedzieć się, jaki użytkownik (nazwa konta) aktualnie pracuje na danej konsoli. Wylogować się z konta root i powrócić do pierwszej konsoli.

Rozwiązanie:
Aktywne były następujące konsole: dradzik tty1 root tty2

1.2.3. Gdzie szukać pomocy?

Nawet, jeśli pamięta się funkcje poszczególnych poleceń, trudno jest jednak przyswoić sobie wszystkie parametry, które są dla nich dostępne. Polecenie man pomoże w

tej kwestii, przez wyświetlenie odpowiednich stron z elektronicznej instrukcji obsługi i podpowiadanie, które polecenia przydadzą się w operacji, jaką mamy zamiar wykonać. Aby wyjść z programu man naciskamy klawisz q

Zadanie 1.2. Sprawdzić, do czego służy polecenie ls wpisując w linii poleceń man ls i jaka opcja umożliwia wyświetlenie plików ukrytych. Spróbuj wykonać polecenie ls /etc

Rozwiązanie: Polecenie "ls" słóży do wyświetlenia zawartości bierzącego katalogu w kolejności alfabetycznej (pomijając pliki i foldery ukryte), aby wyświetlić również pliki i foldery ukryte należy wpisać "ls -a". "ls /etc" wyświetla zawartość folderu etc.

1.2.4. Przerwanie wykonywania poleceń

Gdy zachodzi potrzeba przerwania działania wykonujących się programów można tego dokonać kombinacją klawiszy Ctrl-C.

1.2.5. Dodawanie indywidualnego konta użytkownika

Korzystając z uprawnień superużytkownika można dodać nowe konto użytkownika, na którym będziemy pracować, gdy nie będziemy potrzebować korzystać z uprawnień użytkownika root.

Polecenie adduser w trybie interaktywnym umożliwia dodanie nowego użytkownika. Uwaga, polecenie adduser jest skryptem, który nie jest dostępny we wszystkich dystrybucjach. Do dodawania nowego użytkownika będziemy w przyszłości wykorzystywać polecenie useradd.

Zadanie 1.3. Dodaj swoje indywidualne konto o nazwie postaci jkadzialka dla osoby Jan Kądziołka, czyli bez polskich znaków, zamieniając swoimi danymi dane z przykładu:

```
adduser jkadziolka
```

Zmiana hasła

Do zmiany hasła w dowolnej chwili dla użytkownika np. userX wykorzystuje się polecenie passwd userX.

Wylogowanie użytkownika

Aby sprawdzić czy działa nowo założone konto należy się wylogować poleceniem **logout** i wprowadzić dane z utworzonego indywidualnego konta.

1.2.6. Przejście na konto konto superużytkownika

Może się zdarzyć, że logując się na konto zwykłego użytkownika, będziemy chcieli stać się superużytkownikiem. Normalnie można wylogować się ze zwykłego konta i zalogować jako root. Zamiast tego można skorzystać z polecenia su, aby zalogować się bezpośrednio na konto root, pozostając nadal zalogowanym na konto zwykłego użytkownika. [Ctrl+d] (lub **exit**) pozwoli na powrót do zwykłego konta.

1.2.6.1. sudo - selektywny dostęp do poleceń superużytkownika

Standardowo Linux przyznając dostęp root stosuje podejście „wszystko albo nic". Najczęściej jednak interesuje nas jakaś możliwość pośrednia. sudo pozwala wybranym użytkownikom na uruchomianie określonych poleceń w imieniu superużytkownika, bez podawania hasła root.[1)]

Sudo korzysta z pliku konfiguracyjnego /etc/sudoers, aby stwierdzić, którzy użytkownicy mogą się nim posługiwać i do jakich jeszcze poleceń każdy z nich ma dostęp po uruchomieniu sesji sudo.

Zadanie 1.4. Za pomocą mc wprowadź do pliku konfiguracyjnego sudo dwie linie, zastępując konto jkadziolka swoim kontem:

```
aligeza ALL=(ALL) ALL
jkadziolka ALL=(ALL) ALL
```

Taki wpis oznacza, że ze swojego konta można uruchomić dowolny program na prawach root. Żeby np. ze swojego konta uruchomić mc na prawach root wprowadź polecenie:

```
sudo mc
```

Pytanie o hasło wymaga podania swojego hasła, a nie użytkownika root.

Zadanie 1.5. Korzystając z man lub podanej literatury opisz szczegółowo konfigurację sudo

Rozwiązanie: Konfiguracja sudo (pod rootem) odbywa się na dwa sposoby: poprzez ręczną edycję pliku /etc/sudoers lub domyślnym edytorem vi za pomocą zlecenia: visudo lub sudo visudo -f /etc/sudoers.

Wpisy mają ogólną postać: user1 host=(user2) program czyli: użytkownik user1 może wykonać na komputerze host aplikację o nazwie program z uprawnieniami użytkownika user2.

W praktyce wygląd wpisów może mieć postać:

root ALL=(ALL) ALL zezwala userowi root na uruchomienie za pomocą sudo wszystkich programów w systemie.

nazwausera1 ALL = ALL zezwala userowi nazwausera1 na uruchomienie (za pomocą sudo) wszystkich programów w systemie (także będzie mógł zmienić hasło rootowi!).

nazwausera2 ALL = ALL, !/usr/bin/passwd root zezwala userowi nazwausera2 na uruchomienie (za pomocą sudo) wszystkich programów w systemie oprócz polecenia zmiany hasła dla roota

nazwausera3 ALL = NOPASSWD: /usr/bin/crontab, PASSWD: /usr/bin/procmail, /usr/bin/lpr zezwala userowi nazwausera3 na uruchomienie (za pomocą sudo) tylko trzech programów: crontab, procmail, lpr, z tym, że do uruchomienia zleceń: procmail, lpr potrzebne jest hasło, a do uruchomienia zlecenia crontab nie ma potrzeby podawania hasła.

zezwala userom należącym do grupy users na uruchomienie (za pomocą sudo) komendy passwd, ale bez prawa do modyfikacji hasła usera root.

antek localhost=(root) /sbin/ifconfig zezwala userowi antek na uruchomienie "w imieniu" roota zlecenia ifconfig, wyświetlającego interfejs sieciowy

---

[1)] Za pomocą ról administracyjnych można w bardziej wyszukany sposób dokonać podziału dostępu do konta root.

▷**Rozwiązanie 1.**

1.3. Linuksowy system plików

W Linuksie wszystkie pliki umieszczone są w katalogach, które z kolei są hierarchicznie połączone ze sobą w jedną strukturę plików. Inaczej niż w MS-DOS, w którym każdej partycji odpowiada oddzielna hierarchia, w Linuksie istnieje jedna hierarchia zawierająca wszystkie partycje. Główny katalog drzewa katalogów (root directory) oznaczany jest / (slash-em). Przy uruchamianiu linuxa musi zostać zamontowany system plików z punktem montowania (/) (poleceniem mount podłączamy systemy plików z urządzeń pamięciowych do głównego drzewa katalogów. Pliki odpowiadające poszczególnym urządzeniom znajdują się w katalogu /dev. Montowanie urządzeń może być wykonane jedynie przez superużytkowników). Najczęściej do punktu montowania / przypisuje się partycję z systemem plików Linuxa extn. Pozostałe partycje można zamontować w katalogach, których nazwa zależy od administratora systemu. Katalog korzenia / od którego rozpoczyna się struktura plików w Linuxie, zawiera kilka katalogów systemowych. Katalogi systemowe zawierają pliki i programy używane do uruchomienia i utrzymywania systemu. Katalog korzenia zawiera również katalog o nazwie home, który może z kolei zawierać katalogi macierzyste wszystkich użytkowników systemu. Podczas logowania użytkownik zostaje umieszczony w katalogu macierzystym. Każdy użytkownik posiada swój katalog macierzysty. Bezwzględna ścieżka katalogu macierzystego dla ułatwienia oznaczana jest przez znak tyldy ~.

Plikom można nadawać nazwy, używając znaków alfabetu, podkreślenia oraz cyfr. W nazwie można również umieszczać kropki i przecinki. Nie powinno się jednak rozpoczynać nazwy pliku od cyfry oraz kropki. Inne znaki, takie jak ukośniki, znaki zapytania czy gwiazdki, są zarezerwowane jako znaki specjalne i nie mogą stanowić części nazwy pliku. Nazwy mogą mieć długość do 256 znaków (np. Pierwszy.plik.stworzony). Pliki rozpoczynające się kropką są ukryte.

W Linuksie wszystko oprócz danych i wykonywalnych programów jest plikiem zarówno katalogi, jaki i fizyczne urządzenia. Oznacza to, że istnieją pliki reprezentujące konsole wirtualne (/dev/ttyn), drukarkę(/dev/lpn), CD-ROM(/dev/cdrom), dyski twarde (/dev/hdxn), stacje dyskietek (/dev/fdn) nawet pamięć RAM. Te specjalne pliki noszą nazwę urządzeń i znajdują się w katalogu /dev. Kiedy Linux komunikując się z danym urządzeniem, robi to przez odczytywanie i zapisywanie danych do jednego z plików odpowiadającego wybranemu urządzeniu. Linux do oznaczania ścieżek wykorzystuje zwykły slash (/) podczas gdy MSWindows stosuje w tym celu backslash (\).

Wyświetlenie aktualnego katalogu dokonuje polecenie **pwd**.

Zadanie 1.6. Sprawdź aktualny katalog.

Zmianę katalogu aktualnego realizuje polecenie **cd**.

Przykład 1.1.
```
cd /dev    przechodzi do katalogu z urządzeniami
cd ..      przejście do katalogu nadrzędnego
cd ~       przejście do katalogu macierzystego
cd ../../etc   powoduje 2 razy przejście
               do katalogu nadrzędnego a
               następnie do podkatalogu etc
```

Zadanie 1.7. Spróbuj wykonać powyższe przykłady, za każdym razem sprawdź poleceniem **pwd** aktulany katalog.

1.3.1. Listowanie zawartości katalogów

Polecenie ls listuje zawartości katalogów ls ls . wyświetla informacje o poszczególnych plikach z katalogu roboczego ls .. wyświetla informacje o poszczególnych plikach z katalogu nadrzednego ls –l wyświetla informacje o poszczególnych plikach ls –l | more wyświetla informacje o poszczególnych plikach (w przypadku gdy wszystkie nie mieszczą się na ekranie strona po stronie)

Poleceniem man dowiedzieć się co robią przełączniki –F –R w poleceniu ls.

Zadanie 1.8. Wylistuj wszystkie pliki ukryte z katalogu domowym (symbol gwiazdki oznacza dowolny ciąg wartość, pytajnik dowolny znak. Przykładowo ls we∗k wyświetli pliki o nazwach we1k wewqwqk we22k itp. natomiast ls ws?k wyświetli jedynie we1k)

1.4. Prawa dostępu dla plików i katalogów

Każdy plik i katalog w Linuxie posiada zestaw praw dostępu określających, kto ma dostęp do pliku i jakie ma prawa dostępu. Umożliwia to ochronę niektórych plików przed niepowołanym odczytem, a inne pozwala np. upublicznić. Aby sprawdzić jakie prawa posiadają pliki i katalogi w wybranym liściu drzewa katalogów używamy polecenia ls –l. W pierwszej kolumnie wyświetlone zostaną prawa dostępu (np. d rw– r–– –––), a w pozostałych kolejno liczba łączników, nazwa właściciela, nazwa grupy, rozmiar pliku w bajtach, data i czas ostatniej modyfikacji, nawa pliku (katalogu) . Aby odczytać zezwolenia do danego pliku rozbijamy dziesięć znaków, w których zakodowane są prawa na 4 części. Znak na pozycji 1, dla poziomej kreski oznacza plik (-) , dla katalogu jest nim (d), plik specjalny odpowiadający urządzeniu blokowemu (b), znakowemu (c), semafor (s), współdzielony obszar pamięci (m) . Następne trzy znaki mówią czy właściciel może odczytywać (– nie, r tak), zapisywać (– nie, w tak), uruchamiać (– nie, x tak). Kolejne trzy znaki definiują w ten sam sposób zezwolenie dla konkretnej grupy użytkowników, a ostatnie trzy dla pozostałych użytkowników nie będących jego właścicielem ani członkami wymienionej grupy.

Do przypisania praw dostępu służy polecenie:
chmod parametry_zezwoleń nazwa(y)_plikow_lub_katalogow parametry zezwoleń używane są w kombinacji znaków kogo dotyczą: u-właściciela, g-grupy, o-wszyscy inni sposobu modyfikacji + - przyznanie, – - odebranie typu dostępu r-prawo odczytu, w-prawo zapisu, x-prawo uruchamiania.

Przykład 1.2.

- chmod o−w plik cofa prawo zapisu pozostałym użytkownikom
- chmod g+rx przyznaje prawo odczytu i uruchamiania członkom grupy
- chmod go+x przyznaje prawo uruchamiania członkom grupy i pozostałym użytkownikom.

Jeżeli zamiast pliku podana zostanie nazwa katalogu to zezwolenia będą miały trochę inny sens.

- r−− prawo odczytu oznacza możliwość listownia jego zwartości poleceniem ls, czyli przeglądania jakie pliki znajdują się w katalogu, ale nie ujawnia żadnych atrybutów plików (np. praw własności, rozmiaru);
- −w− prawo zapisu oznacza, iż w danym katalogu można tworzyć i usuwać pliki lub katalogi;
- −−x prawo uruchamiania daje możliwość przejścia do katalogu poleceniem **cd** i pozwala na dostęp (zapis, odczyt, wykonanie) do plików tam zapisanych, jednak wskazanie musi być za pomocą pełnej ścieżki.
- r−x pozwala użytkownikom uruchamiać programy z katalogu i wyświetlać zawartość katalogu, ale nie zezwala na tworzenie i usuwanie plików z tego katalogu.
- −wx Stosowane do katalogów rozwijanych. Użytkownik może przejść do katalogu i umieścić w nim pliki, ale nie może poznać nazw plików, umieszczonych tam przez innych.

Poszczególne prawa dostępu wyrażają odpowiednie liczby z zakresu od 0 do 7 (r-4 w-2 x-1), i tak 0 oznacza brak dostępu, a 7(4+2+1) pełen dostęp.Np polecenie: chmod 777 nauka_linuxa daje pełne przywileje dla wszystkich.

Zadanie 1.9. Znaleźć wartość liczbową, dla której właściciel może zapisać i wykonać, grupa czytać i wykonać, a pozostali tylko wykonać.

Rozwiązanie: Oznaczenie aby właściciel mógł zapisywać i wykonywać , grupa odczytywać i wykonywać, a reszta tylko wykonywać to -wx r-x –x, więć wartość liczbowa tego zapisu to 351.

Do zmiany właściciela pliku (katalogu) przy uprawnieniach root służy polecenie: chown nowy_wlasciciel nazwa pliku

---

Zadanie 1.10. Bity lepkie? ACL w Windows i Linuksie.

Rozwiązanie: Bit lepki oznacza że folder może usunąć tylko jego właściciel. ACS - Access Control List jest to lista kontroli dostępu. Dzięki niej możemy ustalić odczyt, zapis i wykonywanie dla dowolnego użytkownika i dowolnej grupy (a nie tylko dla właściciela i grupy do której on należy)

1.5. Polecenia operujące na plikach i katalogach

Jak wygląda pełne wskazywanie ścieżki?

Tworzenie katalogów

Służy do tego plecenie mkdir.

Zadanie 1.11. Utworzyć katalogi so i so1 w katalogu macierzystym (powinniśmy być zalogowani na kącie użytkownika nie root!).

W tym katalogu so za pomocą polecenia man mkdir, dowiedz się jak przy tworzeniu katalogu można od razu ustawić prawa dostępu. Utworzyć w nim katalogi o nazwach i prawach dostępu,

wynikających z nich tj. −−− r−−− −w− −−x rw− r−x ..., czyli wszystkie kombinacje jakie mogą się pojawić. Grupa oraz pozostali nie mają żadnych praw dostępowych.

Możliwe jest również użycie tworzenie kilku zagnieżdżonych katalogów jednocześnie stosując przełącznik −p.

Zadanie 1.12. Wprowadź: mkdir −p ~/so/bash/skrypty_stratowe. Spowoduje utworzenie wpierw katalogu bash a następnie w nim katalogu skrypty_startowe

Usunięcie katalogu

Polecenie rmdir usuwa pusty katalog.

Zadanie 1.13. Usuń katalog so1

Usunięcie niepustych katalogów można dokonać poleceniem rm. Zostanie ono omówione przy okazji operacjach na plikach.

- **cat** nazwypliku1 [nazwaplikun] łączy pliki podane w argumencie i wypisuje ich zawartość na standardowe wyjście.

  Zadanie 1.14. Wyświetl zawartość pliku /etc/passwd

  Za pomocą polecenia użytego tak: **cat** > nazwa_pliku można utworzyć plik o nazwie nazwa_pliku. Następnie można wprowadzić jego zawartość i je zakończyć sekwencją klawiszy CtrlD.

  Zadanie 1.15. Utwórz plik ~/so/czytaj.txt i wprowadź do niego tekst:

```
#!/bin/bash
echo "Janie,"
echo "nie zapomnij regularnie wykonywać wszystkich zadań z SO-LAB."
echo "Później będziesz wypoczywał."
```

- more nazwa_pliku umożliwia przeglądanie pliku strona po stronie.

  Zadanie 1.16. Wykonaj poprzednie zadanie przy pomocy tego polecenia.

- od - wyświetla zawartość dowolnych plików, wykorzystuje się go najczęściej w odniesieniu do plików, które nie należą do kodu ASCI. Np. można wyświetlić zawartości programu wykonywalnego od −t x /usr/bin/find wypisując bajty pliku w kodowaniu szesnastkowym.

  Zadanie 1.17. Spróbuj wyświetlić jego zawartość poleceniem od, a potem **cat**.

- Często najbardziej interesująca część pliku znajduje się na jego końcu lub początku. Polecenie tail −n plik zwraca −n linii pliku. Wprowadź polecenie:

```
tail -2 /etc/passwd
```

  Zwraca ostatnio dodane konta użytkowników.

- Z kolei polecenie head −n plik zwraca początkowe n linii pliku. Pierwsza linia w pliku skryptowym informuje system operacyjny jakiego programu powłoki użyć do wykonania zawartego w nim kodu. Wyświetl pierwszą linię pliku /etc/init.d/sudo.

- cp służy do kopiowania plików i katalogów.

  Przykład 1.3. cp /etc/passwd /so spowoduje skopiowanie pliku z kontami użytkowników do katalogu so cp /so/passwd passwd1 spowoduje skopiowanie pliku do nowego pliku o nazwie passwd1 gdy wejdziemy do katalogu /so to wprowadzenie polecenia cp passwd passwd1 ../ spowoduje skopiowanie dwóch plików do katalogu nadrzędnego.

Zadanie 1.18. Wykonaj powyższe przykłady. Skopiuj katalog so z całą zawartością do ~/sodel

---

Ogólnie: kopiowane będą wszystkie pliki do katalogu wskazanego ostatnim argumentem.

Ponieważ system nie jest w stanie rozpoznać bez odwołania się do systemu plików, czy dana nazwa reprezentuje plik czy katalog, polecenie cp a b nie jest jednoznaczne. Jeżeli b nie istnieje, nastąpi skopiowanie pliku a do pliku b. Gdy b istnieje i jest zwykłym plikiem, nastąpi jego nadpisanie zawartością pliku a. Jeżeli b istnieje i jest katalogiem, to plik a zostanie skopiowany do katalogu b.

Sprawdź znaczenie przełączników Przełącznik −r i −R.

- rm nazwa_pliku służy do usuwania plików i katalogów.

```
rm -f  usuwa plik bez pytania czy na pewno chce się go usunąć.
rm -r może usuwać katalogi wraz z podkatalogami i z plikami.
```

Zadanie 1.19. Usuń pliki ~/passwd?. Usuń katalog ~/sodel

- mv zrodlo przenaczenie służy do przenoszenia plików.

Zadanie 1.20. `w katalogu so utwórz katalog pliki-konfiguracyjne`
`z katalogu ~ przenieś wszystkie pliki ukryte do stworzonego\` `powyżej k`

1.5.1. Łączniki symboliczne i twarde

Poleceniem ln oryginalna−nazwa−pliku dodatkowa−nazwa−pliku można nadać plikowi oryginalna−nazwa−pliku, który na nośniku informacji elektronicznej zajmuje ustalony obszar pamięci więcej niż jedną nazwę przypisaną do tego samego obszaru. Odnośniki do tego samego pliku lub katalogu można umieścić w kilku różnych katalogach i modyfikacja zawartości w jednym z nim powoduje jednoczesna modyfikację w pozostałych (bo wskazują na ten sam obszar pamięci). Linux nie usunie z systemu pliku, dopóki będą istniały do niego dowiązania. Przedstawiane dowiązanie jest dowiązaniem twardym. Jeśli jednak chcemy utworzyć dowiązanie do katalogu (w przeciwieństwie do pliku), musi to być dowiązanie symboliczne. Jest to również jedyny sposób na utworzenie dowiązania do pliku znajdującego się na innej partycji lub na innym dysku sieciowym.

Polecenie ln zrodlo cel tworzy twardy link o nazwie cel do pliku o nazwie zrodlo. Link wydaje się być kopią oryginalnego pliku, ale w rzeczywistości tylko jedna kopia pliku jest przechowywana, a dwa lub więcej wpisy w katalogu na niego wskazują. Dowolna zmiana w tym pliku jest automatycznie widoczna wszędzie. Kiedy jeden wpis do katalogu jest kasowany, inny (inne) pozostają nienaruszone. Ograniczeniami twardych dowiązań (linków) są następujące:

- pliki muszą znajdować się na tym samym systemie plików,
- twarde linki do katalogów albo do plików specjalnych są niemożliwe.

Zadanie 1.21. Utwórz twardy link do pliku ~so/passwd o nazwie passwdlocal i umieść go w katalogu macierzystym.
Spróbuj teraz usunąć plik ~so/passwd i sprawdź cz istnieje link passwdlocal.

Polecenie ln –s zrodlo cel Tworzy dowiązanie symboliczne (soft link) o nazwie cel do pliku zrodlo. Dowiązanie symboliczne zwyczajnie określa ścieżkę, gdzie znajduje się realny plik. W przeciwieństwie do twardych linków, źródło i cel nie muszą być na jednym systemie plików. W porównaniu do twardych dowiązań, wadami dowiązań symbolicznych są: jeśli oryginalny plik zostaje skasowany, link jest "zepsuty" – wskazuje źródło donikąd; dowiązania symboliczne mogą także tworzyć zapętlone referencje (jak zapętlone referencje w arkuszu kalkulacyjnym lub bazie danych, na przykład "a" oznacza "b" a "b" oznacza "a"). W skrócie, dowiązania symboliczne są wspaniałym narzędziem i bardzo często się je stosuje, mogą one jednak stwarzać dodatkowe problemy.

Zadanie 1.22. W katalogu ~/so utwórz link symboliczny o nazwie etclink wskazujący na katalog /etc
Wejdź do katalogu etclink i przeglądnij jego zawartość poleceniem ls

Zadanie 1.23. Do czego służy przełącznik –i polecenia ls.

Rozwiązanie: "ls -i" powoduje wyświetlenie id każdego pliku/katalogu.

1.6. Programy usługowe

1.6.1. Lokalizacja programów

Wprowadzane polecenia w powłoce użytkownika w większości przypadków zlokalizowane są na dysku w postaci wykonywalnych plików. which nazwa_programu spowoduje wyświetlenie w którym miejscu znajduje się szukany program.

Zadanie 1.24. Dowiedz się gdzie znajduje się program cp.

Rozwiązanie: Program cp znajduje się w bin/cp

Więcej informacji zwraca polecenie whereis oraz locate, które zwracają dodatkowe informacje i innych współtowarzyszących plikach.

Zadanie 1.25. Spróbuj zlokalizować program adduser przedstawionymi w tym pkt. poleceniami. Wyświetl jego zawartość.

1.6.2. Polecenie find

Znalezienie interesujących nas plików pleceniem find to jedno, ale możliwość natychmiastowego skorzystania z nich jest wielkim plusem Linuksa. Dostępny parametr **–exec** powoduje wykonanie na każdym ze znalezionych plików określone polecenie. Polecenie

```
find / -name "*smieci*" -exec mv {} /kosz_na_smieci \;
```

spowoduje znalezienie w całym drzewie katalogów wszystkich plików w których nazwie pojawia się słowo smieci a następnie przeniesie je poleceniem mv do katalogu /kosz_na_smieci. Nawiasy klamrowe {} oznaczają miejsce, w którym powinna znaleźć się nazwa każdego znalezionego polecenia, a kombinacja znaków \; oznacza koniec polecenia.

Zamiast parametru **–exec** można użyć parametr –ok, który będzie żądał potwierdzenia przy każdym wykonaniu funkcji.

Poniżej podano kilka przykładów użycia funkcji find:

- Znajduje wszystkie podkatalogi w katalogu /home find /home **–type** d
- Znajduje pliki, które nie były otwierane przez ostatnie 10 dni find / –atime +10 Pokazuje wszystkie pliki większe od 5M, wraz z opisami dodatkowymi find / –size +5M –ls

Zadanie 1.26. Wyświetl wszystkie nazwy zwykłych plików, które były zmieniane w ciągu ostatniego dnia. Wynik polecenia przekieruj operatorem > do pliku ~\so\1\zmieniniane_2010–10–10 wprowadzając dzisiejszą datę w miejsce 2010–10–10. Utwórz przed wykonaniem programu potrzebne katalogi.

Zadanie 1.27. Dowiedz się jak znaleźć pliki które mają określonego właściciela man. Każdy warunek można zanegować poprzedzając go znakiem !. Np. polecenie find ~ ! –perm –220 znajduje pliki z katalogu domowego, które nie mają ustawionego bitu zapisu dla użytwkownika lub nie mają ustawionego bitu zapisu dla grupy.

Rozwiązanie: Aby znaleźć pliki użytkownika dradzik należy użyć "find / -user dradzik"

1.7. Przekierowania wejścia/wyjścia

Zmiana standardowego wyjścia może być dokonana operatorem >, który powoduje, że wyjście uruchamiananego procesu zostanie skierowane do wskazanego pliku zamiast na ekran **cat** /etc/passwd > ~/nikt przekierowanie standardowego wyjścia do pliku. Wykonaj ten przykład.

Zadanie 1.28. Znajdź we wcześniejszym materiale użycie tego operatora do tworzenia nowego pliku.

Operator < powoduje zmianę standardowego wejścia procesu. Dane zamiast z klawiatury mogą zostać np. pobrane z pliku:

```
cat < /etc/passwd
```

W tym przypadku wejście zostało przekierowane tak, aby dane były pobierane z pliku. Wyjście nie zostało zmienione, więc dane trafiają na ekran. W poniższym poleceniu

```
cat </etc/passwd > ~/passwdtmp
```

przekierowano zarówno wejście i wyjście, tworzy się w efekcie kopia pliku passwd.

Operatory >> umożliwia dopisywanie wyników na końcu istniejącego pliku. Polecenie

```
cat </etc/passwd >> ~/passwdtmp
```

spowoduje powielenie zawartości passwd w istniejącym już pliku. Jeżeli plik passwdtmp by wcześniej nie istniał wówczas działanie operatora >> jest takie jak >.

Operator << ma szczególne znaczenie i powoduje, że do procesu zostaną przekazane dane z bieżącego standardowego wejścia aż do napotkania wskazanego napisu:

```
cat << KONIEC
To może być długa sekwencja
linii i znaków
która może być nawet złożonym programem
KONIEC
```

Dane wpisywane z klawiatury będą w tym przypadku gromadzone i zostaną przekazane do programu **cat** dopiero gdy pojawi się odpowiednia sekwencja inicjująca, u nas KONIEC.

cat2 /etc >/dev/null lub przekierowanie standardowego wyjścia diagnostycznego.

1.8. Potoki

Strumień wyjściowy procesu może być połączony strumieniem wejściowym innego procesu za pomocą *potoku*. Dane produkowane przez pierwsze polecenie są buforowane przez SO i następnie odczytywane przez drugie polecenie. Łączenie odbywa się za pomocą operatora |, np.: ls /etc | more. Katalog /etc może zawierać dużą liczbę plików i polecenie ls wyrzuci je na ekran pozostawiając tylko ostatnią stronę wyników. Żeby je dokładnie przejrzeć potrzebne jest stronicowanie wyniku. Program more umożliwia stopniowe przeglądanie danych pochodzących ze strumienia wejściowego.

Zadanie 1.29. Wprowadź i przeanalizuj następujące przykłady:

```
cat /etc/passwd | head -n 80 | more

cat /etc/group | head -n 3 | tail -n 1

find /usr -name "*.bin" | head -n 4

\vskip0.5cm
\textbf{Rozwiązanie:}
Pierwsza komenda pokazuje 80 pierwszych linii pliku passwd
druga komenda okazuje 3 ostatnie linie pliku group
trzecia komenda znajduje w katalogu usr pliki o dowolnej nazwie z rozszerzeniem bin i wyświetla 1
```

1.9. Filtry

Filtr jest programem, który czyta dane ze standardowego wejścia, przetwarza je i wypisuje wyniki na standardowym wyjściu. Dotychczas poznaliśmy takie programy jak **cat**, tail, head.

1.9.1. Wyszukiwanie tekstu - grep

Filtr grep ma za zadanie wypisanie tylko tych linii, które zawierają określone wyrażenie. Jego składnia jest następująca:

```
grep [opcje] szukane_wyrazenie [lista_plikow]
```

Gdy lista_plikow jest pusta to przeszukiwane jest standardowe wejście.

Przykład 1.4. Proste przeszukanie pliku /etc/fstab w celu znalezienia linii komentarza (komentarze w skryptach bash w pierwszej linii zawierają znak #):

```
grep ^# /etc/fstab
```

gdy po znaku # ma być spacja, wówczas wprowadzamy w cudzysłowie

```
grep "^# " /etc/fstab
```

W wyrażeniach regularnych znak ^ oznacza że linia ma rozpoczynać się od znaków znajdujących się za nim.

Zadanie 1.30. Przeczytać w podręczniku grep punkt o wyrażeniach regularnych.

Zadanie 1.31. Znaleźć w pliku /etc/group do jakich grup należy użytkownik aligeza, czyli linie w których występuje.

Zadanie 1.32. Jak znaleźć tylko te linie, które nie zawierają się w podanym wyrażeniu (man)

Rozwiązanie: "grep -v wzorzec" znajduje pliki które nie zgadzają się z wzorcem.

1.9.2. Wycinanie kolumn poleceniem cut

Filtr cut umożliwia wybieranie z linii tekstu z określonych kolumn. Kolumny tekstu mogą być wybierane na dwa sposoby:

- przez podanie kolumny z wykorzystaniem przełącznika –f, z tabulatorem jako domyślnym separatorem kolumn. Znak oddzielający kolumny można zmienić za pomocą przełącznika –d;
- przez podanie przesunięcia w znakach –c lub bajtach –b.

Przykład 1.5. W pliku /etc/passwd dane w wierszach oddzielone są znakiem :. W pierwszej kolumnie znajduje się nazwa użytkownika. Jej wydobycie można zrealizować w następujący sposób:

```
cut -f1 -d ":" /etc/passwd
```

Zadanie 1.33. Z pliku /etc/fstab wydobądź z każdego wiersza tylko punkty montowania partycji. Pominąć linie komentarza.

ĆWICZENIE 2

Podstawowe narzędzia linuksa

Zadanie 2.1. W katalogu so utwórz katalog 2. W nim utwórz plik o nazwie pltoplain.c i zapisz w nim kod na podstawie programu plconv.c, który polskie litery ze stron kodowych latin i win1250 zamieni na ich spłaszczone odpowiedniki np. literę *ą* zamieni na *a* itd.

Program należy skompilować za pomocą polecenia

```
cc -o pltoplain pltoplain.c
```

▷**Rozwiązanie 1.** [n] Poniższy program bierze ze standardowego wejścia znaki i każdy zamienia na jego spłaszczoną wersje.

```
#include <stdio.h>
#include <string.h>

int main ( argc , argv )
int argc;
char *argv[];
{
char c;
int i;

char pl[]=
{164,143,168,157,227,224,151,141,189,165,134,169,136,228,162,152,171,190};
char brak[]=
{65,67,69,76,78,79,83,90,90,97,99,101,108,110,111,115,122,122};

while ((c=getchar()) != EOF)
{
      for(i=0;i<18;i++) {
                        if(c==pl[i]) c=brak[i];}
                        putchar(c);
}
}
```

sort wyświetl zawartosc pliku uniq sort, tee split paste

2.1. System plików

dd, wykorzystanie do tworzenia pustych plików o określonym rozmiarze, oraz do kopiowania np. bootrekord 512b

Z pliku FS−Hierarchy.pdf przeczytać jakie urządzenia są skojarzone z jakim plikiem

/dev/zero urządzenie wykorzystane do reprezentacji zerowych bajtów. Plik z zawartością samych zerowych bajtów.

dd **if**=/dev/zero of=~/so/2/plikopartycja obs=10M count=1 tworzy plik 10 megabajtowy

mkfs tworzenie systemu plików

mount

du - określa wielkość obszaru zajętego na dysku przez wszelkie rodzaje plików i katalogów. df - podaje ile wolnego miejsca pozostało jeszcze na dysku. time - powoduje wykonanie wyszczególnionego programu i zmierzenie czasu jego wykonania.

Zadanie 2.2. Znaleść 4 największe pliki w katalogu /etc.

Rozwiązanie: ls -l—sort -k5 -n—tail -4.

Zadanie 2.3. Znaleść urzytkowników którzy są właścicielami plików w katalogu home.

Rozwiązanie: ls -l—cut -c 15-22—uniq. Właściciele plików w home to:
1
aligęza
dradzik

Zadanie 2.4. Do czego służą polecenia paste,tee,split?

2.2. Komendy

2.2.1. Paste

Paste:

paste [-s] [-d lista] [–serial] [–delimiters=lista] [–help] [–version] [plik...]
Paste wypisuje na standardowym wyjściu linie składające się z kolejno odpowiadających sobie linii każdego z podanych plików, oddzielonych znakiem tabulacji. Jeśli nie podano żadnych plików lub plik o nazwie '-', używane jest standardowe wejście.

Opcje

–serial
Wkleja linie jednego pliku na raz, a nie naprzemiennie po jednej linii z każdego pliku.

–delimiters=lista
Zamiast tabulacji do oddzielania połączonych linii używa po kolei ograniczników z listy. Gdy lista zostanie wyczerpana, zaczyna na nowo od jej początku.

2.2.2. Tree

Tee:

tee [-a] [-i] [Plik ...]
Tee wczytuje dane ze standardowego wejścia, wypisuje je na standardowym wyjściu i jednocześnie kopiuje do podanych plików. Wyjście nie jest buforowane.
Parametry

-a - dopisuje do pliku, zamiast nadpisywać.
-i - ignoruje sygnały SIGINT.

Zwracane wartości

Tee zwraca następujące wartości na zakończenie:

* 0 - wejście zostało pomyślnie skopiowane do plików.
* ¿0 - wystąpił błąd.

2.2.3. Split

Split:

split - rozdziel plik na kawałki
split [-linie] [-l linie] [–lines=linie] [-b bajty[bkm]] [–bytes=bajty[bkm]] [-C bajty[bkm]] [–line-bytes=bajty[bkm]] [–verbose] [–help] [–version] [plikwej [plikwyj-prefiks]]

Split tworzy jeden lub więcej plików wynikowych (tyle ile potrzeba), zawierających kolejne sekcje z plikwej, lub ze standardowego wejścia jeśli nie podano żadnego lub podano nazwę '-'. Domyślnie split do każdego pliku wynikowego wstawia 1000 linii, albo to co pozostało jeśli jest odeń mniejszy.

Nazwy plików wynikowych składają się z przedrostka, po którym następuje grupa liter, wybrana tak, iż połączenie plików wynikowych posortowanych wedle nazwy daje oryginalny plik

wejściowy, w odpowiednim porządku. Domyślny przedrostek nazwy pliku wynikowego to ‘x’. Jeśli podano argument plikwyj-prefiks, jest on używany zamiennie jako przedrostek nazwy pliku wyjściowego.

OPCJE
-linie, -l linie, –lines=linie
Wstawia linie linii pliku wejściowego do każdego pliku wyjściowego.

-b bajty[bkm], –bytes=bajty[bkm]
Wstaw bajty bajtów pliku wejściowego do każdego pliku wyjściowego. bajty to niezerowa liczba całkowita, plus opcjonalnie jeden z poniższych znaków określających inną jednostkę:

b bloki 512-bajtowe.

k bloki kilobajtowe.

m bloki megabajtowe.

-C bajty[bkm], –line-bytes=bajty[bkm]
Wstawia do każdego pliku wynikowego tyle kompletnych linii pliku wejściowego ile jest możliwe bez przekraczania bajty bajtów. Jeśli istnieje linia dłuższa, wstawia z niej wskazaną liczbę bajtów do każdego pliku wynikowego aż pozostanie mniej niż bajty bajtów linii, potem kontynuuje normalnie. bajty ma taki sam format jak w opcji –bytes.

–verbose Wyświetla komunikat na standardowym wyjściu diagnostycznym przed otwarciem każdego z plików wynikowych.

Zadanie 2.5. Opisać dd,fdisk,mkfs.

2.2.4. DD

dd:

dd - konwertuje plik podczas jego kopiowania

Składnia

dd [opcja]..
dd kopiuje plik (domyślnie ze standardowego wejścia na standardowe wyjście), z wybieranymi przez użytkownika rozmiarami bloków wejścia/wyjścia; podczas kopiowania opcjonalnie wykonuje na nim konwersje.

Opcje

Za tymi z opcji, które przyjmują wartości numeryczne (bajty i bloki) może następować mnożnik oznaczający odpowiednio: xM M// c 1
w 2
b 512
kD 1000
k 1024

MD 1,000,000
M 1,048,576
GD 1,000,000,000
G 1,073,741,824
lub dowolny ze standardowych przyrostków określających rozmiar bloku jak np. ‘k=1024’ (zobacz ‘Rozmiar bloku’ w podręczniku fileutils(1)).

if=plik
Czyta z pliku zamiast ze standardowego wejścia.

of=plik
Zapisuje do pliku zamiast na standardowe wyjście. O ile nie podano conv=notrunc skraca plik do rozmiaru wskazywanego przez seek= (0 bajtów jeśli nie podano seek=).

ibs=bajtów
Odczytuje bajtów naraz.

obs=bajtów
Zapisuje bajtów naraz.

bs=bajtów
Odczytuje i zapisuje bajtów naraz. Przesłania ibs i obs.

cbs=bajtów
Konwertuje bajtów naraz.

skip=bloki
Pomija bloki ibs-bajtowych bloków pliku wejściowego przed kopiowaniem.
seek=bloki
Pomija bloki obs-bajtowych bloków pliku wyjściowego przed kopiowaniem.

count=bloki
Kopiuje tylko bloki ibs-bajtowych bloków wejściowych z pliku wejściowego, zamiast kopiowania wszystkiego aż do końca pliku.

conv=konwersja[,konwersja...]
Konwertuje plik według argumentu konwersja. (Bez spacji wokół przecinków.)
Konwersje:

ascii Konwertuje EBCDIC na ASCII.

ebcdic Konwertuje ASCII na EBCDIC.

ibm
Konwertuje ASCII na alternatywne EBCDIC.

block
Dla każdego wiersza wejścia, wysyła na wyjście cbs bajtów, zamieniając znak nowej linii na spację i w razie potrzeby uzupełniając spacjami.

unblock

Zamienia końcowe spacje w każdym bloku wejściowym wielkości cbs na znak nowej linii.

lcase Zmienia duże litery na małe.

ucase Zmienia małe litery na duże.

swab Zamienia miejscami każdą parę bajtów wejściowych. GNU dd działa inaczej niż inne gdy odczytanych zostanie nieparzysta ilość bajtów. Jeśli plik wejściowy zawiera nieparzystą liczbę bajtów, ostatni bajt jest po prostu kopiowany (ponieważ nie ma go z czym zamienić).

noerror
Kontynuuje po błędach odczytu.

notrunc
Nie obcina pliku wyjściowego.

sync Uzupełnia każdy blok wejściowy do rozmiaru ibs bajtów końcowymi bajtami zerowymi (NUL).

2.2.5. fdisk

fdisk:
fdisk - Obsługa tablicy partycji dla Linuksa

Składnia:

fdisk [-u] [-b rozmiar sektora] urządzenie

fdisk -l [-u] [-b rozmiar sektora] [urządzenie ...]

fdisk -s partycja ...

fdisk -v

Opis

Dyski twarde mogą być podzielone na jeden lub więcej logicznych dysków zwanych partycjami. Podział ten jest zapisany w tablicy partycji znajdującej się w sektorze 0 dysku.

W świecie BSD mówi się o tzw. “plastrach dyskowych” (‘disk slices’) i “etykiecie dysku” (‘disklabel’).

Linux potrzebuje przynajmniej jednej partycji – mianowicie partycji przeznaczonej na główny system plików. Może używać plików i/lub partycji wymiany, ale te drugie są bardziej wydajne. Tak więc, zazwyczaj używana jest druga partycja linuksowa przeznaczona na partycję wymiany (tzw. swap). Na komputerach o architekturze zgodnej z Intelem, BIOS, używany do uruchamiania systemu, często ma dostęp tylko do pierwszych 1024 cylindrów dysku. Z tego powodu osoby mające duże dyski często tworzą trzecią partycję o rozmiarze kilku MB, zazwyczaj montowaną w katalogu /boot, która przechowuje obraz jądra i kilka dodatkowych plików potrzebnych w czasie uruchamiania systemu, żeby mieć pewność, że te rzeczy są dostępne dla BIOS-u. Mogą być różne przyczyny, związane z bezpieczeństwem, ułatwieniem administrowania i tworzenia kopii zapasowych lub testowaniem, żeby używać większej liczby partycji niż to minimum.

fdisk (w pierwszej wersji jego wywołania) to obsługiwany z pomocą menu program do działania na tablicy partycji dysku twardego. Rozumie partycje DOS-owe oraz etykiety dysków typu BSD lub SUN-a.

Urządzenie jest zwykle jednym z:
/dev/hda
/dev/hdb
/dev/sda
/dev/sdb
(/dev/hd[a-h] – to dyski IDE, /dev/sd[a-p] – dyski SCSI, /dev/ed[a-d] – dyski ESDI, /dev/xd[ab] – dyski XT). Nazwa urządzenia odnosi się do całego dysku.

Partycja to nazwa urządzenia, za którą następuje liczba. Na przykład, /dev/hda1 jest pierwszą partycją pierwszego dysku twardego IDE w systemie. Dyski IDE mogą zawierać do 63 partycji, a dyski SCSI – do 15. Zobacz także /usr/src/linux/Documentation/devices.txt.

Etykieta dysku typu BSD/SUN może opisywać 8 partycji, z których trzecia powinna być specjalną partycją oznaczającą cały dysk. Partycji, która używa swojego pierwszego sektora (jak na przykład partycja wymiany) nie należy umieszczać w cylindrze 0, ponieważ zniszczy to etykietę dysku.

Etykieta dysku typu IRIX/SGI może opisywać 16 partycji, z których jedenasta powinna być partycją całego "woluminu", natomiast dziewiąta powinna być tzw. "nagłówkiem woluminu". Nagłówek woluminu także pokrywa całą tablicę partycji, tzn. zaczyna się w bloku zerowym i domyślnie ciągnie się przez pięć cylindrów. Pozostałe miejsce w nagłówku woluminu może być użyte przez wpisy głównych katalogów. Żadna partycja nie może nachodzić na nagłówek woluminu. Także, nie należy zmieniać typu nagłówka woluminu, ani tworzyć na nim systemu plików, ponieważ spowoduje to utratę tablicy partycji. Proszę używać tego typu etykiety dysku tylko podczas pracy z Linuksem na komputerach IRIX/SGI lub podczas pracy z dyskami IRIX/SGI pod Linuksem.

Tablica partycji typu DOS może opisać nieograniczoną liczbę partycji. W sektorze 0 jest miejsce na opis 4 partycji (zwanych podstawowymi -'primary'). Jedna z nich może być partycją rozszerzoną; jest ona jakby pudełkiem zawierającym partycje logiczne, których deskryptory można znaleźć w wiązanej liście sektorów, z których każdy poprzedza odpowiadającą partycję logiczną. Cztery podstawowe partycje, niezależnie od tego czy są obecne, czy nie, otrzymują numery od 1 do 4. Numery partycji logicznych zaczynają się od 5.

W tablicy partycji typu DOS początkowe przesunięcie ('offset') i rozmiar każdej partycji przechowywany jest na dwa sposoby: jako bezwzględna liczba sektorów (zapisywana na 32 bitach) i jako trójka Cylindry/Głowice/Sektory (Cylinders/Heads/Sectors) (zapisywana na 10+8+6 bitach). Pierwszy zapis jest w porządku - przy 512-bajtowych sektorach będzie działał aż do 2 TB. W przypadku drugiego zapisu występują dwa małe problemy. Przede wszystkim pola C/H/S mogą być wypełnione tylko wtedy, gdy znana jest liczba głowic oraz liczba sektorów na ścieżce. Po drugie, nawet jeżeli te liczby są znane, to te 24 bity, które są dostępne, nie wystarczają. DOS używa tylko C/H/S, Windows – obu, a Linux nigdy nie używa C/H/S.

Jeśli jest to możliwe, fdisk automatycznie uzyska informacje o geometrii dysku. Niekoniecznie musi to być fizyczna geometria dysku (co więcej, nowoczesne dyski w rzeczywistości nie mają czegoś takiego jak fizyczna geometria, a w każdym razie nie mają niczego, co mogłoby być opisane w prostej postaci cylindry/głowice/sektory), lecz geometria, której używa MS-DOS dla

tablicy partycji.

Zazwyczaj wszystko idzie dobrze i nie ma żadnych problemów, jeżeli Linux jest jedynym systemem na dysku. Jednakże, jeśli dysk będzie dzielony z innymi systemami operacyjnymi, to dobrym pomysłem jest utworzenie co najmniej jednej partycji fdiskiem pochodzącym z innego systemu operacyjnego. Linux, podczas uruchamiania, przegląda tablicę partycji i stara się wywnioskować, jaka (fałszywa) geometria dysku jest wymagana, żeby mógł dobrze współpracować z innymi systemami.

Gdy drukowana jest tablica partycji, dokonywane jest sprawdzenie konsystencji wpisów tablicy. Sprawdzane jest, czy fizyczne i logiczne punkty startowe i końcowe są takie same, oraz czy partycja rozpoczyna się i kończy w granicach cylindrów (poza pierwszą partycją).

Niektóre wersje MS-DOS tworzą pierwszą partycję, która nie rozpoczyna się na granicy cylindra, lecz na 2 sektorze pierwszego cylindra. Partycje rozpoczynające się na cylindrze 1 nie mogą rozpoczynać się na granicy cylindra, lecz raczej nie powinno to sprawiać problemów, chyba że używasz OS/2.

Funkcje sync() i ioctl() BLKRRPART są wywoływane (o ile zmieniono tablicę partycji) przed wyjściem, kiedy tablica partycji zostanie już zaktualizowana. Dawno temu było konieczne przeładowanie systemu (reboot) po użyciu fdiska. Teraz nie powinno być takiej potrzeby; co więcej, zbyt szybkie przeładowanie systemu może spowodować utratę jeszcze nie zapisanych danych. Proszę zauważyć, że zarówno jądro, jak i dysk mogą buforować dane.

OSTRZEŻENIE DOS 6.x

Komenda FORMAT z DOS 6.x szuka informacji w pierwszym sektorze obszaru danych partycji i traktuje je jako godniejsze zaufania niż informacje z tablicy partycji. Dosowy FORMAT oczekuje od dosowego FDISK-a, że ten wyczyści pierwsze 512 bajtów obszarów danych przy każdej zmianie rozmiarów. Dosowy FORMAT zajrzy do tych dodatkowych danych nawet z flagą /U – uważamy to za błąd tych programów.

Tak więc używając cfdiska lub fdiska do zmiany rozmiaru partycji dosowej, należy też użyć dd do wyzerowania pierwszych 512 bajtów tej partycji przed użyciem dosowego FORMAT. Na przykład, po utworzeniu dosowej partycji programem cfdisk, powinno się wykonać polecenie “dd if=/dev/zero of=/dev/hda1 bs=512 count=1”, które zeruje pierwszych 512 bajtów partycji.

BĄDŹ NIEWYOBRAŻALNIE OSTROŻNY przy używaniu komendy dd, gdyż mała pomyłka może spowodować zniszczenie wszystkich danych z dysku.

Dla najlepszych wyników, zawsze powinieneś używać fdiska specyficznego dla danego systemowi operacyjnemu. Na przykład, partycje dosowe powinieneś tworzyć dosowym FDISK-iem, a linuksowe – linuksowym fdiskiem lub cfdiskiem.

Opcje

-b rozmiar sektora
Określa rozmiar sektora dysku. Możliwe wartości są następująca: 512, 1024 lub 2048. (Ostatnie jądra znają rozmiar sektora. Proszę używać tej opcji tylko dla starych jąder lub w celu nadpisania wartości podanej przez jądro).

-l

Drukuje tablice partycji dla podanych urządzeń i kończy działanie. Jeżeli nie podano żadnych urządzeń, to używane są urządzenia wymienione w pliku /proc/partitions (o ile taki istnieje).

-s partycja Wypisuje na standardowym wyjściu rozmiar partycji (blokach).

-v Drukuje numer wersji fdiska i kończy działanie.

Błędy

Istnieje kilka różnych programów *fdisk. Każdy z nich ma swoje problemy i zalety. Prosimy wypróbować ich w następującej kolejności: cfdisk, fdisk, sfdisk. (Istotnie, cfdisk jest pięknym programem, mającym surowe wymagania co do tablic partycji, które akceptuje, tworzącym tablice partycji wysokiej jakości. Należy go używać, jeżeli tylko jest taka możliwość. fdisk jest programem mającym wiele błędów, robiącym mętne rzeczy – zazwyczaj zdarza mu się dać sensowne wyniki. Jego pojedynczą zaletą jest obsługa etykiet dysków typu BSD i innych nie-DOS-owych tablic partycji. Należy go unikać, jeżeli tylko jest taka możliwość. sfdisk jest programem tylko dla hakerów – interfejs użytkownika jest okropny, ale program jest poprawniejszy od fdiska i potężniejszy od zarówno fdiska, jak i cfdiska. Co więcej, może być używany nieinterakcyjnie).

Etykiety dysku typu IRIX/SGI nie są jeszcze obsługiwane przez jądro. Co więcej, katalogi główkowe (header directories) IRIX/SGI nie są jeszcze w pełni obsługiwane.

Brak opcji “zapisz tablicę partycji do pliku”.

2.2.6. mkfs

mkfs:
mkfs - zbuduj linuksowy system plików

Składnia

mkfs [-V] [-t typ fs] [opcje-fs] filesys [bloki]

Opis

mkfs służy do budowania linuksowego systemu plików na zadanym urządzeniu, zwykle partycji dysku twardego. filesys jest albo nazwą urządzenia (np. /dev/hda1, /dev/sdb2) albo punktem montowania (np. /, /usr, /home) dla danego systemu plików. Parametr bloki jest liczbą bloków, jakie zostaną użyte dla zadanego systemu plików.

mkfs zwraca kod zakończenia równy 0 w przypadku powodzenia, zaś 1 przy porażce.

W rzeczywistości mkfs jest po prostu interfejsem dla różnych poleceń budujących systemy plików (mkfs.typ fs) dostępnych w Linuksie. Polecenie budowania odpowiednie dla danego typu systemu plikowego wyszukiwane jest w wielu różnych katalogach jak /sbin, /sbin/fs, /sbin/fs.d, /etc/fs, /etc (dokładna lista definiowana jest na etapie kompilacji, ale zawiera co najmniej /sbin i /sbin/fs), a ostatecznie w katalogach wymienionych w zmiennej środowiskowej PATH. Przeglądnij, proszę, strony podręcznika właściwe dla poleceń budowania konkretnych typów systemów plików. Znajdziesz tam więcej szczegółów.

Opcje

-V Tryb gadatliwy. Wyświetlaj szczegółowe komunikaty, łącznie ze wszystkimi wywoływanymi poleceniami specyficznymi dla systemu plików. Podanie tej opcji więcej niż jednokrotnie zabrania wykonywania jakichkolwiek poleceń specyficznych dla danego systemu plików. Naprawdę przydatne tylko do testowania.

-t typ fs Określa typ tworzonego systemu plików. Jeśli nie podano, używany jest domyślny (obecnie ext2).

opcje-fs
Opcje specyficzne dla danego systemu plikowego, jakie mają zostać przekazane do faktycznego polecenia tworzenia systemu plików. Mimo, że nie jest to gwarantowane, poniższe opcje obsługiwane są przez większość poleceń tworzących systemy plików.

-c Sprawdź uszkodzone bloki (bad blocks) urządzenia przed utworzeniem systemu plików.

-l nazwapliku Odczytaj listę błędnych plików z pliku o podanej nazwie.

-v (verbose) Włącz tryb gadatliwy.

Błędy

Wszystkie opcje ogólne muszą poprzedzać i nie mogą być łączone z opcjami specyficznymi dla systemu plików. Niektóre z programów budujących pewne systemy plików nie obsługują opcji -v ani nie zwracają znaczących kodów zakończenia. Ponadto, niektóre z programów tworzących systemy plików nie rozpoznają automatycznie wielkości urządzania i wymagają podania parametru bloki.

Zadanie 2.6. W jaki sposób można zamontować partycje fatową w systemach windows?

Rozwiązanie: Montowanie dysków (partycji) pod Windows wydaje się czymś dziwnym, ale to jest naturalna operacja w systemie plików NTFS. Nie wiem czy była dostępna w Windows NT 4, ale na pewno jest w 2000, XP i Viście. A banalna do przeprowadzenia i od razu ułatwia parę rzeczy. Ja na przykład teraz trzymam gry (niestety te zajmują coraz więcej miejsca na dysku – to się w głowie nie mieści ile!) i dane multimedialne (zdjęcia, filmy, mp3) na oddzielnych partycjach, na drugim dysku.

Konsola zarządzania dyskami

Ale są one podmontowane jako odpowiednie katalogi na jednym z dysków, więc mam szybki dostęp i nie mam bałaganu z literami dysków. A parę godzin po dołożeniu do komputera drugiego dysku twardego mogło zacząć brakować liter w alfabecie. Swoją drogą zawsze zastanawiała mnie taka sytuacja – co wtedy Windows robi?

Aby podmontować partycję (która moze być sformatowana zarówno w FAT, jak i NTFS) potrzebujemy innej partycji, sformatowanej w systemie NTFS i pustego folderu. Oraz nieocenionego narzędzia “Zarządzanie dyskami”, dostępnego z konsoli “Zarządzanie komputerem”, bądź bezpośrednio przez wywołanie diskmgmt.msc.

Klikając prawym klawiszem myszy na partycji mamy możliwość zmiany litery dysku lub ścieżki.

Nic nie stoi na przeszkodzie, aby jedna partycja była zamontowana w kilku miejscach i do tego posiadała jeszcze literę dysku.

Zmiana litery dysku i dodawanie ścieżki

Wybieramy “Zainstaluj w następującym pustym folderze plików NTFS”, wybieramy folder i gotowe. Mamy partycję widzianą w Eksploratorze jako katalog. Oznaczony odpowiednią ikoną, dla odróżnienia od normalnych folderów.

Partycje w Eksploratorze

Mechanizm ten, podobny do montowania dysków znanego z systemów Uniksowych, pozwala na posiadanie dużej ilości partycji i zmniejszenie wśród nich bałaganu.

Niestety, jest jedna, całkiem poważna wada. Wyobraźmy sobie, że chcemy zainstalować program na dysku C:, ale brakuje miejsca. Czy coś szkodzi podłączyć dodatkową partycję pod – dajmy na to – podkatalog w “Program Files” żeby mieć więcej miejsca i żeby zachować ład w programach? Niestety, mechanizmy obliczania ilości wolnego miejsca na partycji nie sumują tej ilości z wolnym miejscem na podmontowanych partycjach, co powoduje, że program nadal może sądzić, że miejsca ma za mało. Wielka szkoda.

2.2.7. stab

stab:

fstab - statyczna informacja o systemach plików

Składnia

include ¡fstab.h¿

Opis

Plik fstab zawiera opisowe informacje na temat różnych systemów plików. fstab jest wyłącznie czytany przez programy i nie jest przez nie zapisywany. Prawidłowe stworzenie tego pliku i zarządzanie nim jest obowiązkiem administratora. Każdy system plików jest opisywany przez osobną linię; pola w każdej linii są oddzielane przez znak spacji lub tabulacji. Kolejność rekordów w pliku fstab jest istotna, ponieważ fsck(8) , mount(8) i umount(8) w celu wykonania swoich zadań dane z pliku fstab pobierają sekwencyjnie.

Pierwsze pole, (fs spec), opisuje specjalne urządzenie blokowe lub zdalny system plików przeznaczony do zamontowania.

Drugie pole, (fs file), wskazuje na miejsce, w którym ma być zamontowany dany system plików. Dla partycji wymiany (ang. “swap partition”) to pole powinno zawierać wartość “none”.

Trzecie pole, (fs vfstype), opisuje typ systemu plików. Obecnie system obsługuje poniższe rodzaje systemów plików:

minix lokalny system plików, obsługujący nazwy plików o długości 14, lub 30 znaków.

ext lokalny system plików z dłuższymi nazwami plików i większymi iwęzłami. Ten system plików został zastąpiony systemem plików ext2 i nie powinien już być używany.

ext2 lokalny system plików z dłuższymi nazwami plików, większymi iwęzłami i wieloma innymi dodatkami.

xiafs lokalny system plików z dłuższymi nazwami plików, większymi iwęzłami i wieloma innymi dodatkami.

msdos lokalny system plików dla partycji MS-DOS.

hpfs lokalny system plików dla partycji HPFS.

iso9660 lokalny system plików używany przez napędy CD-ROM.

nfs system plików służący do montowania partycji z systemów zdalnych.

swap partycja dysku używana jako przestrzeń wymiany.

Jeżeli vfs fstype jest ustawiony jako “ignore”, to pole jest ignorowane. Jest to przydatne do wskazania, które partycje dysku są aktualnie nieużywane.

Czwarte pole, (fs mntops), opisuje opcje montowania związane z danymi systemami plików.

Jest sformatowane jako lista opcji, oddzielonych przecinkami. Zawiera co najmniej typ montowania oraz dodatkowe opcje, odpowiednie do danego typu systemu plików. Aby uzyskać dokumentację dostępnych opcji dla nie-nfs'owych systemów plików, zobacz mount(8) . Aby uzyskać dokumentację wszystkich opcji specyficznych dla nfs, zobacz nfs(5) . Wspólną opcją dla wszystkich typów systemów plików jest opcja “noauto” (nie montuj gdy podano “mount -a”, np. podczas ładowania systemu), i “user” (pozwala użytkownikowi na montowanie). Aby dowiedzieć się więcej, zobacz mount(8) .

Piąte pole, (fs freq), jest używane przez komendę dump(8) aby wykryć, który system plików musi być odłączony. Jeżeli nie ma informacji o tym polu, zwracana jest wartość 0 i dump przyjmuje, że dany system plików nie musi być odłączany.

Szóste pole, (fs passno), jest używane przez program fsck(8) aby zadecydować, jaka powinna być kolejność sprawdzania systemów plików podczas ładowania systemu. Główny system plików powinien mieć fs passno równą 1, zaś inne systemy plików powinny mieć fs passno równe 2. Systemy plików w obrębie dysku będą sprawdzane po kolei, natomiast

systemy plików na różnych dyskach będą sprawdzane jednocześnie, aby wykorzystać wielozadaniowość udostępnianą przez sprzęt. Jeżeli nie podano szóstego pola, zostaje zwrócona wartość zero i fsck zadecyduje, że dany system plików nie musi być sprawdzany.

Aby poprawnie czytać pola z fstab należy używać getmntent(3) .

Pliki

/etc/fstab Plik fstab jest przechowywany w /etc.

Zadanie 2.7. Czym jest powłoka?

Rozwiązanie:
Powłoka systemowa (ang. shell) - program komputerowy pełniący rolę pośrednika pomiędzy systemem operacyjnym lub aplikacjami a użytkownikiem, przyjmując jego polecenia i "wyprowadzając" wyniki działania programów. To pośrednictwo nie jest obowiązkowe (programy mogą być bardziej "samodzielne").

Powłoka często sama zawiera podstawowe polecenia, gdy jednak wydane przez użytkownika polecenie nie jest wbudowane, uruchamiany jest program zewnętrzny. Po zalogowaniu użytkownik znajduje się w linii poleceń i może wydawać polecenia systemowi.

Zadanie 2.8. Szyfrowanie Plików.

Aby zaszyfrować pliki hasłem należy użyć komendy gpg -c nazwa pliku i podajemy hasło. Aby odszyfrować pliki używamy komendy gpg nazwa pliku.

ĆWICZENIE 3

Powłoka BASH

Czym jest powłoka

Uaktywnienie powłoki BASH. grep bash /etc/passwd

Załóż konto o loginie 1 i hasle 1 zmien domyslna powłokę na wywołanie progrmu mc

Przełączyć się na 2 konsole i zalogowac się jak 1

Nie pracujemy w XWindowsach, ponieważ sprawdzian będzie przez ssh w trybie tekstowym!!!.

whereis bash

wieloprogramowość program echo

readline **bind** –P **bind** –l klawisze skrótów, przypisanie poleceń. w windows jak globalnie przypisać klawisze skrótów

3.1. Struktura skryptu

3.1.1. Informacje o procesach

/proc man **ps** **ps** –axu &, **ps**, **kill** sleep (sleep 60;who)&

nice szeregowanie procesów.

at wykona o określonym czasie polecenie.

3.1.2. Pierwszy skrypt

Ma za zadanie wyświetlenie wszystkich użytkowników zalogowanych do systemu i zapisanie ich loginów do pliku ~/so/4/logins w strukturze:

```
2010-01-12 12:30
root
aligeza
...
2010-01-13 10:00
aligeza
...
```

Rozwiązanie:

```
#! /bin/bash
date >> lista
who | cut -c 1-8 >> lista
```

Stwórz w katalogu ~/bin plik o nazwie listazalogowanych. Nadaj mu atrybut, że może być wykonywany przez wszystkich.

- Interpreter poleceń skryptu. Pierwsza linia skryptu o tym informuje #! /bin/bash. Dlaczego w większości skryptów ma wpis #!/bin/sh
- # - znak komentarza w skrypcie.
- Wykonanie skryptu (zmienna PATH)

3.2. Zmienne powłoki

s.76

Definiowanie zmiennej: nazwazmiennej=wartosc, usunięcie zmiennej **unset** nazwazmiennej

```
echo "$katalogSO"
echo "Zmienna \$katalogSO ma wartosc \"$katalogSO\""
echo 'Zmienna \$katalogSO ma wartosc \"$katalogSO\"'
```

3.3. Parametryzacja skryptów

```
#!/bin/sh
echo "Argument 1=$1"
echo "Argument 2=$2"
echo "Argument 3=$3"
echo "$0"
echo "Liczba argumentow $#"


listazalogowanych 12 aa "2110-10-12 10:45"
```

Odwołanie się do argumentów od 10 wzwyż wymaga użycia wbudowanej komendy interpretera **shift**, która powoduje przeniesienie wartości zmiennych pozycyjnych na pozycje o jeden mniejszy.

```
#!/bin/sh
echo "Argument 1=$1"
shift
echo "Argument 2=$1"
shift
echo "Argument 3=$1"
```

3.4. if

```
if warunek
then
 instrukcje gdy prawdziwy warunek
else
 instrukcje gdy fałszywy warunek
fi
```

Warunkiem instrukcji **if** może być dowolne polecenie. Brany jest pod uwagę status zakończenia tego polecenia. Każdy proces po zakończeniu swojej pracy zwraca do systemu status zakończenia, który jest liczbą od 0 do 255. Status 0 oznacza prawdę pozostałe fałsz, jest to inna konwencja od języka C.

3.4.1. Polecenie test

Służy do testowania różnego rodzaju warunków.

```
#!/bin/sh
if test "$1" = "n"
then
 polecenie do zarchiwizowania obecnego pliku loginu
 i utworzenia nowegoecho
else
 przepisz już posiadane polecenie
fi
```

Specyfikując warunek do sprawdzania przez polecenie **test** należy pamiętać o tym, że składowe tego warunku muszą być oddzielnymi argumentami tego polecenia, a więc muszą być odseparowane od siebie spacjami. Zapis **if test** $1=n jest błędny!!!

Polecenie **test** ma swój odpowiednik w postaci programu [ ]

```
#!/bin/sh
if [ $1 = n ]
then
 echo "wprowadziles n"
else
 echo "nie tworze nowego"
fi
```

Nawiasy kwadratowe nie służą do wydzielenia warunku, ale są zleceniem uruchomienia programu [. Oznacza to, że przed i po nawiasach muszą wystąpić spacje. Zapis **if** [$1 = n] jest błędny!!!

Spróbuj zapisać poprzedni skrypt i uruchomić bez podania żadnego argumentu. Będzie **if** [= n] **if** ["$1" = "n"]. Zatem, jeżeli wartość może być niezdefiniowana, to należy ją umieścić w cudzysłowie dla spełnienia wymagań polecenia **test**.

3.4.2. Rodzaje testów

man **test** Porównanie łańcuchów tekstów, liczb, plików, test złożone

test - sprawdzanie typów plików i porównywanie wartości

test zwraca kod zakończenia 0 (prawda) lub 1 (fałsz) zależnie od wyniku ewaluacji wyrażenia warunkowego ´wyrażenie´. Wyrażenia mogą być jednolub dwuparametrowe. Każda część wyrażenia musi być osobnym argumentem.

WyraŻenia Warunkowe

Zwróć uwagę na to, że nawiasy powinny być chronione (np. odwrotnym ukośnikiem) przed interpretacją przez powłokę.

Sprawdzanie typu plików
-b plik
Prawda, jeżeli plik istnieje i jest urządzeniem blokowym. -c plik
Prawda, jeżeli plik istnieje i jest urządzeniem znakowym. -d plik
Prawda, jeżeli plik istnieje i jest katalogiem. -f plik
Prawda, jeżeli plik istnieje i jest zwykłym plikiem. -h plik
-L plik
Prawda, jeżeli plik istnieje i jest dowiązaniem symbolicznym. -p plik
Prawda, jeżeli plik istnieje i jest nazwanym łączem (named pipe).
-S plik

Prawda, jeżeli plik istnieje i jest gniazdem (socket). -t [fd]
Prawda, jeżeli plik o deskryptorze fd jest otwarty na terminalu. Jeżeli fd nie jest podane, jego wartość przyjmowana jest jako 1 (standardowe wyjście).

Sprawdzanie praw dostępu
-g plik
Prawda, jeżeli plik istnieje i ma ustawiony bit set-group-id. -k plik
Prawda, jeżeli plik ma ustawiony bit "sticky". -r plik
Prawda, jeżeli plik istnieje i może być czytany. -u plik
Prawda, jeżeli plik istnieje i ma ustawiony bit set-user-id. -w plik
Prawda, jeżeli plik istnieje i można do niego pisać. -x plik
Prawda, jeżeli plik istnieje i może być wykonany. -O plik
Prawda, jeżeli plik istnieje i jego właścicielem jest użytkownik o numerze równym aktualnemu efektywnemu UID. -G plik
Prawda, jeżeli plik istnieje i należy do grupy o numerze równym efektywnemu GID.

Sprawdzanie właściwości plików
-e plik
Prawda, jeżeli plik istnieje.
-s plik
Prawda, jeżeli plik istnieje i ma rozmiar większy niż zero. plik1 -nt plik2
Prawda, jeżeli plik1 jest nowszy (zgodnie z datą modyfikacji) niż plik2.
plik1 -ot plik2
Prawda, jeżeli plik1 jest starszy niż plik2. plik1 -ef plik2
Prawda, jeżeli plik1 i plik2 mają te same numery urządzenia oraz i-węzła, tj. są wzajemnymi twardymi dowiązaniami.

Sprawdzanie łańcuchów znakowych
Poniższe opcje sprawdzają właściwości łańcuchów. Dla test łańcuchy nie są cytowane, choć może zachodzić potrzeba takiej ochrony przed interpretacją przez powłokę znaków o specjalnym dla niej znaczeniu, np. spacji. -z łańcuch
Prawda, jeżeli łańcuch ma długość zero. [-n] łańcuch
Prawda, jeżeli długość łańcucha jest różna od zera. łańcuch1 = łańcuch2
Prawda, jeżeli łańcuchy są jednakowe. łańcuch1 != łańcuch2
Prawda, jeżeli łańcuchy nie są jednakowe

Testy numeryczne
argument1 OP argument2
OP jest może być jednym z niżej wymienionych: -eq, -ne, -lt, -le, -gt lub -ge.
Powyższe dwuargumentowe operatory arytmetyczne zwracają prawdę, jeżeli argument1 jest odpowiedni równy, nierówny, mniejszy niż, mniejszy lub równy, większy niż albo większy lub równy w stosunku do argumentu2. argument1 oraz argument2 mogą być liczbami całkowitymi ze znakiem (lub bez znaku) albo specjalnym wyrażeniem -l łańcuch, którego wartością jest długość łańcucha. Na przykład:
test -1 -gt -2 echo yes
=¿ yes
test -l abc -gt 1 echo yes

=¿ yes
test 0x100 -eq 1
error–¿ test: integer expression expected before -eq

Operatory logiczne
(wyrażenie)
Prawda, jeżeli wyrażenie jest prawdziwe. ! wyrażenie
Prawda, jeżeli wyrażenie jest fałszywe. wyrażenie1 -a wyrażenie2
Prawda jeżeli obydwa wyrażenia są prawdziwe. wyrażenie1 -o wyrażenie2
Prawda jeżeli przynajmniej jedno z wyrażeń jest prawdziwe.

3.5. while

Pętla while jest kolejną użyteczną pętlą występującą we wszystkich językach programowania. Najpierw sprawdza warunek czy jest prawdziwy, jeśli tak to wykonana lista poleceń zawartych wewnątrz pętli, gdy warunek stanie się fałszywy pętla zostanie zakończona.
Składnia:

```
while [ warunek ] ; do
polecenie
done
```

Przykład:

```
#!/bin/bash
x=1;
while [ $x -le 10 ] ; do
echo "Napis pojawił się po raz: $x"
x=$[x + 1]
done
```

Sprawdzany jest warunek czy zmienna x o wartości początkowej 1 jest mniejsza lub równa 10, warunek jest prawdziwy w związku z czym wykonywane są polecenia zawarte wewnątrz pętli: echo "Napis pojawił się po raz: $x" oraz x=$[x + 1], które zwiększa wartość zmiennej x o 1. Gdy wartość x przekroczy 10, wykonanie pętli zostanie przerwane.

3.6. for

Pętla for jest jedną z tych rzeczy, która stanowi o potędze Basha - bo nie w każdym języku można iterować po wynikach jakichś komend czy nawet po liniach w pliku.

Iterować
Wykonywać polecenia zawarte wewnątrz pętli, na każdym składniku listy (argumencie).

W Bashu istnieją jej dwie odmiany:
Pętla for podobna do języka C (C like for)
Składnia

for ((¡inicjacja_zmiennej¿; ¡warunek_zakonczenia¿; ¡zwiększanie/zmniejszanie¿)) ; do
...polecenia w pętli...
done

Pętla ta służy do operacji gdzie znamy ilość elementów z których musimy skorzytać lub do policzenia czegoś, wygenerowania liczb:

Przykład

#!/bin/sh

for ((i=1; $i ¡= 10; i++)) ; do
echo " Iteracja nr: $i"
done

Uwaga:

Ta wersja pętli for nie jest dostępna we wszystkich implementacjach Basha można ją częściowo korzystając z polecenia seq.

Przykład

#!/bin/sh

for i in ‘seq 1 10‘
do
echo "Iteracja nr: $i" done

Oba skrypty wyprodukują identyczne wyjście:

Iteracja nr: 1
Iteracja nr: 2
Iteracja nr: 3
. . .
Iteracja nr: 10

Bardziej bashowa

Inny typ pętli for jest znacznie częściej uzywany, jego składnia przedstawia się następu-

jąco:
Składnia

for ¡zmienna¿ in ¡lista¿ ; do
zrób_coś
i_jeszcze_coś
done

¡lista¿ może być łańcuchem znakowym, tablicą lub po prostu listą słów:

Przykład

for x in jeden dwa trzy ; do
echo "To jest $x"
done

Wynik:

To jest jeden
To jest dwa
To jest trzy

Jak to działa:

Zmiennej x przypisana jest lista, która składa się z trzech elementów: jeden, dwa, trzy. Wartością zmiennej x staje się po kolei każdy element listy, na wszystkich wykonywane jest polecenie:

echo "To jest $x".

Pętla for jest bardzo przydatna w sytuacjach, gdy chcemy wykonać jakąś operację na określonym zbiorze plików np na wszystkich plikach w danym katalogu (lub tylko na tych o określonej nazwie).

Przykład - zamiana nazw plików pisanych DUŻYMI literami na nazwy pisane małymi literami: **

#!/bin/bash
for nazwa in * ; do mv "$nazwa" ‘echo "$nazwa" — tr '[A-Z]' '[a-z]'‘
done

Za zmianę DUŻYCH liter na małe (i na odwrót) odpowiedzialne jest polecenie tr.

3.7. Zmienne specjalne

$0, $#, $@, $*, $?, $$, $!

3.8. Dostosowanie środowiska

Pliki odczytywane przez powłokę

- . bash_profile — podczas zalogowania ~/. profile przeczytaj komentarz. W przypadku braku pliku wykonywany jest /etc/ profile . **source** .bash_profile Zmiany pliku konfiguracyjnego zadziałają bez potrzeby zalogowania i wylogowani
- .bash_logout — wylogowania
- bashrc — uruchomiania nowej powłoki.

aliasy **alias** nazwa=polecenie **alias** lf='ls −F' np.

3.8.1. Zmienne wbudowane

- Zmienne symbolu zachęty PS1="\u−−>". Sprawdzić bieżący znak zachęty.
- PATH, separatorem poszczególnych ścieżek znak :. Dopisanie do ścieżki PATH=$PATH":/home/aligeza/bin" Konie trojańskie.PATH="/home/aligeza/bin:$PATH" - kolejność zmieniona.
- CDPATH, HOME

ĆWICZENIE 4

Samba

4.1. Wstęp

Nie będę się rozpisywał na temat, który jest już po raz n-ty wałkowany (google prawdę Ci powie), więc przedstawię po krótce, czym jest Samba. Samba jest zbiorem aplikacji korzystających z protokołu SMB (Server Message Block). Aby zapewnić działanie podstawowa wersja składa się z dwóch programów smbd oraz nmbd, które należą do rodziny "oprogramowania o otwartym źródle", (Open Source Software, OSS).
Smbd jest demonem odpowiadającym za współdzielenie plików i drukarek oraz zapewniającym uwierzytelnianie. Nmbd to demon, który zapewnia mechanizmy WINS (Windows Internet Name Service), czyli usługi nazewnicze, potrzebne do przeglądania zasobów sieci. Pozostałe aplikacje pakietu samby:
smbclient – klient umożliwiający podłączenie się do udziałów samby lub windows.
smbtar – program archiwizujący
testparm – program testujący plik konfiguracyjny smb.conf
nmblookup – sprawdza nazwy NetBIOSowe
smbpasswd – podstawowy program zarządzający kontami w sambie
smbstatus – wyświetla obecny stan połączeń z sambą
testprns – program sprawdzający czy drukarki są rozpoznawane przed sambe
swat – program umożliwiający zarządzanie sambą poprzez interfejs przeglądarki internetowej

Istnieje jeszcze wiele programów współpracujących z sambą w wielu czynnościach, jednak zawęziłem spis do podstawowych aplikacji. (do góry)

4.2. Instalacja

Sambę pobierzesz z jej domowej stronki www.samba.org

Źródełka
Zanim zaczniesz instalować z źródeł sambe należałoby sprawdzić czy mamy właściwe wersje bibliotek wymaganych przez pliki wykonywalne. Dla bibliotek konsolidowany dynamicznie samba może potrzebować odpowiednich wersji bibliotek. ldd – sprawdza, jakie biblioteki potrzebuje samba dla wersji 3.0.14a:

debian: ldd /usr/local/samba/sbin/smbd
libcrypt.so.1 =¿ /lib/libcrypt.so.1 (0x40022000)
libresolv.so.2 =¿ /lib/libresolv.so.2 (0x4004f000)
libnsl.so.1 =¿ /lib/libnsl.so.1 (0x40061000)
libdl.so.2 =¿ /lib/libdl.so.2 (0x40077000)
libc.so.6 =¿ /lib/libc.so.6 (0x4007a000)
/lib/ld-linux.so.2 =¿ /lib/ld-linux.so.2 (0x40000000)

archiwum gz,bz

debian: tar -zxvf samba-3.0.x.tar
debian: cd samba-3.0.x
debian: ./configure –prefix=/usr/local
debian: ./make make install

Rpm

mdk: rpm –checksig samba-3.0.x.src.rpm
mdk: rpm -ihv samba-3.0.x.src.rpm
- przebudowanie (mdk: rpm –rebuild samba-2.2.x.src.rpm)
- aktualizacja (mdk: rpm -Uhv samba-2.2.x.iy86.rpm)

4.2.1. Reinstalacja

Przed ponowną instalacją należy zabezpieczyć pliki konfiguracyjne smb.conf* - gwiazdka oznacza wszystkie pliki konfiguracji smbpasswd - baza danych użytkowników dotychczasowych samby

Pozbywamy się starych wersji:

rm -rf /usr/local/samba
rm -rf /etc/samba

dla rpm:

rpm -qa — grep samba
rpm -e –nodeps samba
rpm -e –nodeps samba-docs
rpm -e –nodeps samba-client

Pozostawiamy tylko paczki, których nazwa bądź człon nazwy mają yast2 lub kde.

4.2.2. Uruchomienie

Są dwa sposoby na uruchomienie samby. Poprzez inetd lub jako deamon: (np. z /usr/local/etc/rc.d/samba.sh). Jeśli sieć nie przekracza 40 użytkowników można użyć prostszej metody uruchomienia procesów samby z inetd. Do pliku /etc/services dopisz następując wiersze:

vi /etc/services
.....
netbios-ns 137/tcp NETBIOS Name Service
netbios-ns 137/udp NETBIOS Name Service
netbios-dgm 138/tcp NETBIOS Datagram Service
netbios-dgm 138/udp NETBIOS Datagram Service
netbios-ssn 139/tcp NETBIOS Session Service
netbios-ssn 139/udp NETBIOS Session Service
netbios-ssn 445/tcp NETBIOS Session Service
.....

Należy dopisać do inetd.conf

netbios-sss stream tcp nowait.30000 root /usr/local/samba/sbin/smbd smbd
netbios-ns dgram udp wait.400 root /usr/local/samba/sbin/nmbd nmbd
swat stream tcp nowait.400 root /usr/local/samba/sbin/swat swat

(do góry)

4.3. Konfiguracja

Jeśli instalacja przebiegła bez problemów, można przystąpić do konfiguracji. Dostrojenie serwera do zamierzonych celów to długa, kręta droga! Postaram się jak najprościej przedstawić 3 tryby działania samby:

1. Samba dla gości – bez autoryzacji
2.Samba z autoryzacją użytkowników
3.Samba jako PDC (domena NT)

4.3.1. Samba dla gości

Samba dla gości – bez autoryzacji (samba share)
Brak autoryzacji to z pozoru tylko wpuszczenie użytkownika bez całkowitej kontroli. Najpierw musimy przygotować konto użytkownika w Linuksie, na które będą się logować przybysze z sieci, dla których nie będzie miało to rządnego znaczenia, jakiego usera przedstawiają.
Standardowo używam konta o nazwie nobody.

dodanie grupy i użytkownika do systemu, bez dostępu do powłoki:

debian: addgroup nogroup
debian: useradd -c "konto bez autoryzacji" -g nogroup -d /dev/null -s /bin/false nobody

konfiguracja smb.conf – głównego pliku konfiguracyjnego samby:

global
workgroup = siec nazwa grupy roboczej
netbios name = serwer-sieci nazwa serwera w grupie roboczej
server string = Samba server oznaczenie, komentarz serwera
guest account = nobody konto logujacych sie bez autoryzacji
security = share usługa ma jedno hasło dla wszystkich
local master = yes serwer będzie główną lokalna przeglądarką

udzial
path = /tmp/udzial ścieżka do katalogu
guest ok = yes dostęp także dla gości guest only = yes każdy użytkownik traktowany jako gość
read only = no nie tylko do odczytu
printable = no oznaczenie zasobu, przeciwieństwo drukarki
valid user = nobody jedynie uzytkownik nobody może widzieć zasób
force user = nobody zapis w udziale z prawami usera nobody

Katalog, który służy do przedstawiania udziału musi mieć właściciela "nobody", który jest naszym publicznym kontem dla gości w sambie, jeśli nie ustawimy odpowiedniego właściciela nie będziemy mieli prawa do zapisu – udział będzie blokowany zgodnie z systemem plików unix'ów.

debian: chown nobody /tmp/udzial

4.3.2. Samba z autoryzacją urzytkownika

Samba z autoryzacją użytkownika
Autoryzacja użytkownika w sambie wymaga założenia kont w systemie passwd oraz w sambie smbpasswd, istnieje synchronizacja tych dwóch baz danych użytkowników, którą przedstawie poniżej.

dodanie grupy i użytkownika do systemu, bez dostępu do powłoki:

debian: addgroup smbgroup

debian: useradd -c "konto smb" -g smbgroup -d /dev/null -s /bin/false waldek

dodanie użytkownika do bazy samby:

smbpasswd -a waldek

konfiguracja smb.conf

global
workgroup = siec nazwa grupy roboczej
netbios name = serwer-sieci nazwa serwera w grupie roboczej
server string = Samba server oznaczenie, komentarz serwera
guest account = nobody konto logujacych sie bez autoryzacji
security = userzabezpieczenie na poziomie użytkownika
local master = yes serwer będzie główną lokalna przeglądarką

homes
comment = katalog domowy komentarz
browseable = no udział nie rozgłaszany w listach przeglądania
writable = yes udzial nie jest przeznaczony tylko do odczytu

Udział home nie powinien się mieszać z katalogami domowymi użytkowników systemu.

4.4. Samba jako PDC

Samba jako PDC – czyli kontroler domeny NT
Konfiguracja samby, jako PDC wymaga założenia dla każdego stanowiska w sieci działającego jako klient i podłączonego do naszej domeny specjalnego konta, (tzw. konta zaufania) dla systemów z rodziny NT.
Konto jest zwykłym kontem użytkownika naszego Linuksa z kilkoma ograniczeniami –

brak dostępu do powłoki – brak dostępu do innych usług ftp. mail itp.
Ponadto samba musi być główną przeglądarką lokalną, w sieci musi być serwer nazw WINS (także samba lub Windows *NT) Ostatnia czynność to przygotowanie konta administratora, które będzie nam potrzebne do podłączania "końcówek" do domeny.

zakładamy grupę i konto zaufania dla klienta:

debian: addgroup smbmachines
debian: useradd -c "konto maszyny" -g smbmachines -d /dev/null -s /bin/false maszyna1

debian: smbpasswd -a -m maszyna1

Jeśli nie czujesz się na siłach ciągle wpisywać maszyny do bazy użytkowników zautomatyzuj ten proces:
add machine script = /usr/sbin/useradd -d /dev/null -g smbmachines -c 'konto maszyny
passwd chat debug = no
passwd chat = *New*UNIX*password*
*passwd:*all*authentication*tokens*updated*successfully*
passwd program = /usr/bin/smbpasswd -L -a

następnie zakładamy grupę i konto użytkownika:

debian: addgroup smbgroup
debian: useradd -c "konto smb" -g smbgroup -s /bin/false waldek

dodanie użytkownika do bazy samby:

smbpasswd -a waldek

konfiguracja smb.conf:

global
workgroup = siec nazwa grupy roboczej
netbios name = serwer-sieci nazwa serwera w grupie roboczej
server string = Samba server oznaczenie, komentarz serwera
guest account = nobody konto logujących się bez autoryzacji
security = user także konieczne w kontrolerze domeny
local master = yes serwer będzie główną lokalna przeglądarką
domain master = yes serwer będzie kontrolerem domeny
prefered master = yes samba będzie główna przeglądarką

announce as = NT server samba będzie udawać serwer NT

encrypt passwords = yes szyfrowanie haseł
smb passwd file = /etc/samba/smbpasswd ścieżka do pliku z kontami

homes
comment = Katalog domowy
browseable = No udział nie rozgłaszany w listach przegladania
writable = Yes udział nie jest przeznaczony tylko do odczytu

Wersja Home nie jest przygotowana do pracy w sieci z kontrolerem domeny. Natomiast systemy z rodziny NT (winXP) aby podłączyć do domeny należy wprowadzić pewną zmianę w rejestrze, można to zrobić za pomocą programu "regedit".

HKEYLOCALMACHINESYSTEMCurrentControlSetServicesnetlogonparameters RequireSignOrSeal=dword:00000000

Można to także zrobić z poziomu Local Group Policy wyłączając (Disabled) Security Options — Domain Member — Digitally encrypt or sign secure channel data.
(Będziemy musieli przez to zrezygnować z kilku opcji Windowsa)

4.5. Samba i profile

Samba i profile
Profile to jedna z zalet budowy sieci SMB. Pomogą nam zautomatyzować archiwizację, także użytkownicy już nie będą musieli się martwić o kopie bezpieczeństwa, a i nam w niektórych przypadkach ułatwi to życie. Profile nazywane profilami mobilnymi mają jeszcze jedną ważną opcję, w momencie kiedy użytkownik zmieni stanowisko pracy wraz z poprawnym zalogowaniem na komputer, który obecnie obsługuje – dostanie wszystkie swoje stare ustawienia z komputera poprzedniego, pulpit, dokumenty, nawet menu start – mowa oczywiście o systemie Windows. Aby uruchomić poprawnie opcje profili należy w pliku konfiguracyjnym umieścić:

logon path = Profile dynamiczna ścieżka dla profili, dla Windows NT/XP/2000/2003

logon home = profile dynamiczna ścieżka dla profili dla Windows 95/98/98SE/Me

Format profili systemu Win9x oraz 2k,XP... jest inny, dlatego "powinien" być przechowywany w innym katalogu, ścieżka profili posiada format UNC, więc można skierować profile na całkowicie inny serwer.
Pamiętaj o prawach dostępu do katalogu, oczywiście trzeba być członkiem domeny NT.

Profile
path = /home/Profile
browseable = No
writable = Yes

profile
path = /home/profile
browseable = No
writable = Yes

4.6. Netlogon

Netlogon:
Jest udziałem dla skryptów ładowania automatycznego w domenie NT. Jak wiadomo skrypty zawarte w netlogon ładują się automatycznie w chwili poprawnego załogowania się do domeny, możemy dzięki temu ustawić synchronizację czasu, a także automatyczne mapowanie udziałów czy drukarek, istnieje jeszcze wiele opcji, o których postaram się przynajmniej wspomnieć.

Konfiguracja udziału netlogon:

netlogon
comment = Network Logon Service komentarz
path = /var/lib/samba/netlogon
guest ok = no
writable = no
browseable = no

Fizycznie do tego udziału już możemy wrzucić pliki z rozszerzeniem .bat które będą się uruchamiać po zalogowaniu do domeny, należy zwrócić uwagę na jedną rzecz – skrypt .bat powinien być utworzony w systemie windows, ponieważ linux w dokumentach tekstowych nie używa "powrotu karetki" jedynie początek nowej linii, natomiast windows obydwie funkcje.

Aby użyć funkcji serwera czasu należy w smb.conf wpisać jeden wiersz:

time server = yes YES uruchamia serwer czasu

Przykład skryptu logowania:

Skrypt startowy windows
echo Ustawiam aktualny czas

net time apollo-serwer /set /yes
echo Mapuje stacje sieciowe na udziały serwera Archiwum
net use m: apollo-serwerdane

4.7. Bezpieczeństwo

Bezpieczeństwo:
Jeśli wogóle można uważać, że siec oparta na protokole SMB może być bezpieczna, to tyle o ile musimy zabezpieczyć przynajmniej w jakimś stopniu naszą sieć.

Podstawy bezpieczeństwa w smb:

Adresowanie:
Jeśli w sieci mamy 28 userów możemy podzielić ją maską bitową. Zazwyczaj osoba zakładająca sieć korzysta z klasy C zaznaczając w niej maskę o wartości 255.255.255.0 – oznacza to że w tej sieci może być zaadresowanych 253 komputery i każdy będzie widział wszystkich swoich "sąsiadów", dlaczego 253? Bo jeden adres to adres sieci, a drugi to adres broadcast (rozgłoszeniowy). Maską możemy podzielić te 255 adresów na 4 podsieci, ponieważ to nie jest dokument o adresowaniu przedstawię tylko gotowy przykład:

adres sieci: 192.168.0.33
pula adresów użytecznych od 192.168.0.34 do 192.168.0.62
broadcast 192.168.0.63
maska podsieci 255.255.255.224

Tak oto uzyskujemy minimalną pule adresów dla 28 komputerów – w tym momencie zostają nam 2 adresy użyteczne, dla innych komputerów nasza sieć nie będzie widoczna.

Opcje samby:

allow hosts = lista hostów wymienia komputery które mogą się łączyć z sambą, synonim opcji hosts allow.
available = warotść logiczna jeśli jest ustawiona na NO, zabrania dostępu do udziału.

global
bind interfaces only = wartość logiczna jeśli jest ustawiona na YES, listy przeglądania będą udostępnione tylko poprzez interfejsy z listy opcji interfaces.

create mask = liczba ósemkowa określa max. prawa dostępu do nowo tworzonych plików od 0 do 0777

deny hosts = lista hostów synonim opcji hosts deny, określa liczbę komputerów, które nie mogą nawiązywać połączeń z serwerem.

dont descent = lista przecinkowa katalogów nie pozwala przeglądać ani przejść do katalogów.

global
encrypt passwords = wartość logiczna ustawiona na Yes uruchamia szyfrowanie haseł metodą LM/NTLM1/2, wymaga smbpasswd.

force user = nazwa użytkownika smb ustawia nazwę użytkownika korzystającego z udziału.

hosts allow = lista hostów synonim opcji allow hosts, określa listę komputerów które mają dostęp do udziału, adresy ip dozwolone: 0.0.0.0/24

global
interfaces = lista interfejsów ustawia listę interfejsów przez które samba będzie odpowiedać na żądania.

invalid users = lista użytkowników lista userów którzy nie mogą korzystać z udziału.

locking = wartość logiczna Stosuje blokady plików, ustawiona na NO akceptuje żądania przyznania blokady

machine password timeout = sekundy ustawia okres między zmianami hasła komputera w domenie Windows NT. Wartość domyślna to 1 tydzień, czyli 604 800 sekund.

max connections = liczba określa max. liczbę połączeń z udziałem

only guest = wartość logiczna wymusza aby użytkownik udziału znajdował sie na liście username.

read only = wartość logiczna mówi o trybie udziału tylko do odczytu

revalidate = wartość logiczna jeśli jest ustawiona na YES użytkownicy będą za każdym razem musieli wprowadzać hasło przeglądając udział.

global
secutiry = wartość określa sposób autoryzacji: share, user, domain, server.

share modes = wartość logiczna jeśli jest ustawiona na YES samba obsłurzy Windowsowego sposobu blokowania plików w trybie odmowy.

global
socket address = adres IP określa adres, pod którym należy czekać na połączenia. Domyślnie Samba czeka na połączenia pod wszystkimi adresami.

valid user = lista lista użytkowników którzy mogą korzystać z udziału.

veto files = lista ukośnikowa lista plików które nie będą pokazywane klientowi podczas listowania zawartości katalogu.

wide links = wartość logiczna jeśli jest ustawiona na YES, samba podąża za dowiązaniami symbolicznymi i na zewnątrz bieżącego katalogu.

Oprogramowanie dodatkowe:
Dodatkowym sposobem zabezpieczenia nie tylko samego serwera, czy komunikacji sieci, ale także plików w całej sieci SMB jest zastosowanie oprogramowania antywirusowego. Instalacja odbywa się na serwerze gdzie pracuje Samba.

Ochrona antywirusowa z wykorzystaniem ClamAV:
Clam jest to program ochrony antywirusowej działający na systemach *nixowych.Pomostem pomiędzy Sambą a ClamAV będzie samba-vscan. Autorem programu jest Rainer Link a prace nad projektem zapoczątkowane zostały w połowie maja 2001. samba-vscan jest rodzajem filtra, który operuje na wirtualnym systemie plików Samby - VFS (ang. Virtual File System). VFS to nic innego jak zbiór udogodnień dodanych do już istniejącego systemu plików. Wprowadza między innymi dodatkowe wywołania; operacje na katalogach (opendir, readdir, mkdir, itd) i plikach (open, read, close, write, itd).

F-secure:
F-Secure Ant-Virus for Samba Servers jest oprogramowaniem, które zatrzymuje wirusy systemu Linux, Windows, wirusy plików i makr infekujące dokumenty Microsoft Office oraz innego typu złośliwe kody. Oprócz rozpoznawania znanych wirusów rozwiązanie wykrywa również nowe i nieznane na podstawie pewnych cech zawartości typowych dla wirusów. Produkt pozwala na szybką reakcję na infekcje, dzięki możliwości wyspecyfikowania potencjalnie groźnych plików (na bazie nazwy lub rozszerzenia) i ich blokowaniu lub usuwaniu.

Automatyczna ochrona w czasie rzeczywistym
F-Secure Anti-Virus przejmuje wszystkie żądania użytkownika i skanuje zawartość plików zanim te zostaną przesłane adresatowi. Skanowanie zapobiega rozsyłaniu wirusów z serwera. Oprogramowanie F-Secure skanuje również pliki przechowywane na serwerze w poszukiwaniu wirusów oraz złośliwych kodów wykonywalnych. Ochrona wszystkich zasobów serwera jest skuteczna i nie wydłuża czasu pobierania plików z serwera.
Administrator jest powiadamiany o wszystkich istotnych alarmach. Informacje te pozwalają na szybką reakcję. Program posiada wbudowany moduł raportujący, który może powiadamiać administratora o zaistniałych problemach. Administrator może zdefiniować katalogi i pliki (określając typ oraz rozmiar) podlegające skanowaniu lub nie wymagające analizy antywirusowej. Program posiada również w szereg opcji pozwalających na podjęcie działań w przypadku zainfekowanych plików.

SambaSecure Antivirus:
SambaSecure Antivirus to system ochrony plików działający w czasie rzeczywistym dla serwerów Samba. Katalogi i pliki (także skompresowane) współdzielone w sieci są zabezpieczane przed wirusami, robakami i trojanami. Jeżeli plik jest linkiem symbolicznym mechanizm antywirusowy gwarantuje również ochronę pliki oryginalnego, bez względu na jego położenie. SambaSecure Antivirus zapewnia skanowanie na żądanie o zaplanowanej porze lub od razu. Działanie tego rozwiązania nie obejmuje jedynie katalogów dzielonych. SambaSecure Antivirus wykrywa i eliminuje złośliwy kod, który przedostał się na serwery w dowolny sposób. Obsługa SambaSecure Antivirus jest prosta dzięki bezpiecznej konsoli web do zdalnego zarządzania (HTTPS). Po instalacji dostęp do niej może odbywać się z dowolnego punktu w sieci - wystarczy przeglądarka internetowa i dane do autoryzacji. Np. po wykryciu zagrożenia można z konsoli zarządzania przenieść niebezpieczny plik do kwarantanny. Do czasu jego wyleczenia, nikt w sieci nie będzie mógł go otworzyć i nie zarazi swojej stacji roboczej złośliwym kodem.

Bezpieczne połączenie w sieci (przesyłanie haseł):
Microsoft zapewnia nam 4 sposoby zabezpieczenia uzyskania tożsamości użytkownika, są to:

Protokołu Kerberos v5
Certyfikatu klucza publicznego
Protokółu SSL/TLS (Secure Sockets Layer/Transport Layer Security)
Protokółu LM/NTLM-1/2
Obecna wersja samby tj. 3.0 obsługuje wszystkie metody przekazywania haseł przez siec (szyfrowane/nieszyfrowane). Istnieją 3 opcje szyfrowania haseł w protokole SMB:
1.LM (Lanman)
2.NTLM (NT1)
3.NTLM2 (NT2)
Wykorzystane funkcje mieszające to: MD4, MD5, SHA-1, oraz algorytmy symetryczne: DES, 3DES, RC4.

Windows
Przesyłanie haseł niezaszyfrowanych przez sieć stanowi wielkie niebezpieczeństwo dla wszystkich klientów i ich systemy operacyjne w sieci. Obecnie istnieje możliwość przystosowania wszystkich wersji Windowsa do szyfrowania haseł. Jednak standardowo nie wszystkie są bezpośrednio przygotowane:

Workgroup 3.11 – niezaszyfrowane
Windows 95 – niezaszyfrowane
Windows 98 – zaszyfrowane (LM)
Windows Milenium – zaszyfrowane(LM)
Windows NT po S.P.3 – zaszyfrowane (LM/NTLM)
Windows 2000 – zaszyfrowane (LM/NTLM)

Windows XP – zaszyfrowane (LM/NTLM/NTLM2)
Windows 2003 – zaszyfrowane (LM/NTLM/NTLM2)

Aby przygotować Windows'y do pracy w trybie szyfrowania należy wprowadzić w plik rejestru pewne zmiany:

dla windows 95/98/Me

HKEYLOCALMACHINESystemCurrentControlSetServicesVxDVNETSUP EnablePlainTextPassword=dword:00000000

dla windows NT po S.P.3

HKEYLOCALMACHINESYSTEMCurrentControlSetServicesRdrParameters EnablePlainTextPassword=dword:00000000

dla windows 2000/XP

HKEYLOCALMACHINESYSTEMCurrentControlSetServicesLanmanWorkStationParameters EnablePlainTextPassword=dword:00000000

Systemy windows zawsze korzystają z jednej bazy danych, gdzie ukrywają swoje hasła. W Windowsach 9x są to pliki PWL, natomiast 2k/Xp są to pliki SAM.
Wartość 0 dword w rejestrze oznacza szyfrowanie haseł, natomiast wartość 1 brak szyfrowania.

Dsclient to program który pozawala nam na użycie algorytmu NTLM2 dla systemów operacyjnych w wersji Windows 9x/Me. Po zainstalowaniu tego programu który należy pobrać ze strony Microsoftu ewentualnie z nośnika płyty instalacyjnej systemu Windows 2003 z katalogu x:clientswin9xdsclient.exe
Po udanej próbie możemy wybrać jedną z opcji szyfrowania konfigurując rejestr systemowy.

Windows 9x
HKEYLOCALMACHINESYSTEMCurrentControlSetControlLMCompatibilityLevel
wartość dword: 0 – dla haseł LM/NTLM
wartość dword: 3 – tylko dla haseł NTLM2

Windows 2k/Xp/2003
HKEYLOCALMACHINESYSTEMCurrentControlSetControlLsaLMCompatibilityLevel
warotść dword: 0 – dla haseł LM/NTLM
warotść dword: 1 – dla systemu NTLM2 – jeśli sieć obsługuje ten sposób autentyfikacji
warotść dword: 2 – dla LM nie jest stosowane, możliwe tylko NTLM

warotść dword: 3 – dla NTLM2 (wyłącznie)
warotść dword: 4 – LM zostaje odrzucone
warotść dword: 5 – serwer tylko przyjmuje NTLM2

Linux
Plik smb.conf pozwala nam na zmianę opcji szyfrowania. Mamy do wyboru także wiele trybów pracy jak w Windowsach. Jedynym warunkiem jest wersja min. 3.0, kiedy piszę ten artykuł najnowszą wersją jest 3.0.14a.

Dostępne opcje:

lanman auth = yes (domyślnie) samba wysyła hasła LM i NTLM

lanman auth = no serwer korzysta wyłącznie z NTLM

ntlm auth = yes + lanman auth = no serwer korzysta z NTLM/NTLM2

ntlm auth = no + lanman auth = no serwer korzysta jedynie z NTLM2

Opcje dla klienta sieci SMB smbclient:

client lanman auth = yes smbclient korzysta wyłącznie z LM

client lanman auth = no smbclient korzysta z NTLM/NTLM2

client ntlmv2 auth = no smbclient nie będzie wysyłał haseł NTLM2

client ntlmv2 auth = yes smbclient łącząc się będzie wysyłał tylko hasła NTLMv2.

client plaintext auth = yes smbclient będzie wysyłał jedynie nie szyfrowane hasła.

client plaintext auth = no smbclient nie będzie wysyłał haseł nie zaszyfrowanych.

Firewall
Wielu ludzi używa firewalli do blokowania portów programów, których standardowo nie używają albo stanowią zagrożenie dla systemu. Należy pamiętać że protokół SMB nadaje na wielu portach, dla przypomnienia:
Port 135/TCP - used by smbd
Port 137/UDP - used by nmbd
Port 138/UDP - used by nmbd
Port 139/TCP - used by smbd
Port 445/TCP - used by smbd (obsługa DFS – rozproszonego systemu plików)

Nowe wersje
Być może to najważniejsza dziedzina bezpieczeństwa. Błędy oprogramowania na nowo są wykrywane, luki które mogą zaszkodzić poprawnemu działaniu serwera, bąć nawet przejąć nad nim kontrolę, mogą występować w obecnej wersji twojego serwera plików. Najlepszym lekarstwem jest systematyczne zaglądanie na stronę domową samby pod adresem www.samba.org gdzie można dowiedzieć się o wykrytych lukach, błędach i ewentualnie pobrać aktualizację.

4.8. Drukarki pod Sambą

Drukarki pod sambą:
Samba może pracować oczywiście jako centrum druku z wieloma dodatkowymi funkcjami statystycznymi. Od wersji 3.0.2 samba jest wyposażona w bezpośredni interfejs do systemu CUPS (Common Unix Printing System), który dzisiaj jest głównym standardem obsługi drukarek w systemach *nix. Aby skonfigurować serwer wydruku dla naszej sieci najpierw musimy zainstalować drukarkę w systemie, można to zrobić na kilka sposobów, ja pokaże jedynie sposób instalacji bez powłok graficznych czyli w bash

Programem którym zainstalujemy drukarke na USB będzie lpadmin. Z poziomu root'a wpisujemy następujące polecenia:

```
debian: lpadmin -p dj945c -E -P /usr/share/cups/model/HP/DeskJet_940C-cdj970.ppd.gz\\
\\
debian: lpadmin -p dj940 -v usb://dev/usb/lp0\\
\\

Można także lpadmin'em skonfigurować port LPT:\\
\\
debian: lpadmin -p x4510 -E -P /usr/share/cups/model/Xerox/Dokuprint_450-hpijs.pd.gz\\
\\
debian: lpadmin -p x450 -v parallel:/dev/lp0\\
\\

Konfiguracja do drukarki sieciowej:\\
\\
debian: lpadmin -p dj840 -E -P /usr/share/cups/model/HP/DeskJet_840C-cdj840.ppd.gz\\
\\
debian: lpadmin -p dj840C -v socket://192.168.0.2:9100\\
\\
```

Składnia polecenia:
-p – zainstaluj drukarkę
-E – aktywuj drukarkę
-P – użyj następującego sterownika
-v – definicja portu

Przygotowanie druku w sieci:
założenie katalogu i ustawienie odpowiednich praw:

debian: mkdir /var/spool/samba
debian: chmod 777 /var/spool/samba

Konfiguracja smb.conf:

[global]
printing = cups ustawienia interfejsu druku
printcap name = cups określa ścieżkę do pliku z parametrami drukarek, tutaj sprawę załatwia system CUPS

printers
path = /var/spool/samba ścieżka do katalogu gdzie będą trzymane kolejki druku.
browseable = no
guest ok = yes
writable = no
printable = yes
printer admin = root, @ntadmin administratorzy drukarek

Powyższa konfiguracja nakazuje sambie wyświetlenie wszystkich drukarek jakie są zdefiniowane w systemie CUPS jako oddzielne drukarki sieciowe. Istnieje możliwość wyświetlenia w otoczeniu sieciowym drukarek które mają być udostępniane, uzyskuje się to poprzez wpisanie w nazwie udziału modelu drukarki.

Automatyczny dobór sterowników dla windows – sposób samby:
Samba, posiada funkcję doboru sterownika dla systemów łączących się z drukarką, znacznie poprawia dynamiczność dużych sieci. Poniżej sposób konfiguracji tych opcji:

printer driver = nazwa sterownika drukarki ustawia łańcuch, który jest przekazywany klientowi kiedy ten pyta jakiego sterownika drukarki ma użyć w celu przygotowania pliku do wydruku.

global
printer driver file = ścieżka profilu określa położenie pliku msprint.def używanego przez Win9x.

printer driver location = ścieżka sieciowa określa położenie sterownika drukarką, domyślnie serwerPRINTER.

Tworzymy udział dla sterowników drukarek:

global
printer admin = root konto administratora drukarki w sambie.

PRINTER
path = /etc/samba/drivers
guest ok = no
browseable = yes
writable = yes
write list = root
admin users = root, @ntadmin

Założenie katalogów dla różnych systemów w udziale samby:

/W32X86 sterowniki dla Windows NT/2000/XP x86
/WIN40 sterowniki dla Windows 95/98/Me

Sterowniki postscriptowe: Cups:
Każda drukarka może mieć zainstalowany standardowy sterownik postscriptowy, a tłumaczeniem danych na format drukarki zajmie się w tym przypadku Cups. Standard ten dysponuje szerokim zakresem funkcji, ale obsługuje tylko Windows NT, XP, 2003. Zaczniemy od ściągnięcia sterowników postscriptowych dla Windows z serwera projektu CUPS:

```
wget ftp://ftp.cups.org/pub/cups/windows/cups-samba-1.1.16.tar.gz\\
```

Kiedy zakończymy pobieranie i zainstalujemy w katalogu znajdują się potrzebne później sterowniki:

cupsdrvr.dll
cupsui.dll
cups.hlp

Adobe:
Sterowniki Adobe są o wiele trudniejsze do zainstalowania, nie występują w postaci archiwa a jedynie jako plik wykonywalny exe. W tym przypadku musimy zainstalować program na Windows. W zależności od systemu pobieramy sterowniki:

```
Win9x/Me\\
http://download.adobe.com/pub/adobe/printerdrivers/win/4.x/drivers/aps426eng.exe\\
\\
WinNt,Xp,2000\\
http://download.adobe.com/pub/adobe/printerdrivers/win/1.x/winsteng.exe\\
\begin{lstlisting}
\\
Jedynie w momencie instalacji możemy pozyskać sterowniki adobe, w chwili instalacji znajdują się
\\
ADOBEPS4.DRV\\
ADFONTS.MFM\\
ADOBEPS4.HLP\\
DEFPRT2.PPD\\
ICONLIB.DLL\\
PSMON.DLL\\
```

```
\\
Wszystkie te pliki powinny zachować wielki rozmiar znaków w nazwie.\\
Następnie przenieś je do linuksowego katalogu:\\
\\
\begin{lstlisting}
/usr/share/cups/drivers\\
```

Następnie poszukaj pliku ADOBEPS5.DLL i z katalogu w którym go znajdziesz skopiuj pliki ADOBEPS5.DLL ADOBEPSU.DLL i ADOBEPSU.HLP do tego samego katalogu ze sterownikami:

/usr/share/cups/drivers

W tym momencie brakuje nam jedynie specjalnego zasobu dla sterowników, powyżej już przedstawiłem właściwą konfiguracje udziału.

Istnieje jeszcze jedno narzędzie do automatycznego przygotowania udziału drukarek ze sterownikami i drzewa katalogów (dla każdego rodzaju Windows), aby wyeliminować ręczną konfigurację, zrobimy to programem:

cupsaddsmb -a

Po tym momencie zostaniesz poproszony o podanie hasła root do serwera Samby. Jeśli autoryzacja nastąpiła pomyślnie powinieneś ujrzeć w katalogu /etc/samba.drivers nowe katalogi WIN40 W32X86, pierwszy zawiera sterowniki do windows 9x/Me w drugim 32bitowe sterowniki Windows. Dodatkowo znajdziesz tam sterowniki wszystkich drukarek zdefiniowanych w CUPS z rozszerzeniem .ppd. Jedynie zostało nam uruchomić ponownie demona Samby.

Limity zadań drukarek:
Dodatkową ciekawą opcją po uruchomieniu automatycznego instalowania drukarek jest opcja limitów. Można ustawić m.in. liczbę zadań druku klienta o określonym przedziale czasu
całkowitą objętość zadań druku w danym okresie
zapobiec drukowaniu prywatnych dokumentów na drukarce kolorowej
przypisać koszty druku do poszczególnych wydziałów.
Przykłady poleceń:

limit 20 stron tygodniowo na użytkownika
lpadmin -p dj940c -o job-quota-period=604800 -o job-page-limit=20

wolumen druku dla drukarki do 2MB dziennie
lpadmin -p dj940c -o job-quota-period=86400 -o job-k-limit=2048

Wadą tego systemu jest to, że odnosi się ona do wszystkich użytkowników jednocześnie (globalnie). Dodatkowo CUPS nie uwzględnia błędów podczas drukowania.

Dodanie sterowników producenta sprzętu do udziału:

Logując sie do w Windows na konto administratora drukarek (root), dodamy sterowniki do udziału dla różnych systemów operacyjnych rodzinki Windows.
Następnie przechodzimy do katalogu Drukarki i Faksy, ważne jest aby rzeczywiście przejść do tego katalogu, a nie jedynie zaznaczyć drukarkę, którą samba wyświetli w zestawieniu zasobów, wywołujemy menu kontekstowe zainstalowanej drukarki i wybieramy properties(właściwości).
W tym momencie pojawi się pytanie: czy chcesz zainstalować nieistniejący sterownik drukarki? Odpowiadamy No (nie). Następnie uruchamiamy przycisk New Driver (nowy sterownik) na karcie Advanced (dodatkowe opcje). Zaznaczamy odpowiedni sterownik do naszej drukarki na wyświetlonej liście. Kliknięcie finish (gotowe) zakończy instalacje sterownika, ale nie lokalnie, a na serwerze samba. Tak zainstalowaliśmy sterowniki do WinNT,XP,2000.
Aby uruchomić kreatora instalacji drukarki na Win9x/Me przechodzimy na kartę sharing (udział). Może się zdążyć że wcześniej będziemy musieli aktywować funkcję udostępniania. Zaznaczamy opcję Win95/98/Me i zatwierdzamy, Windows poprosi nas o podanie ścieżki docelowej do plików sterowników. Miejmy nadzieje że to bezpośrednio rozpakowane archiwum. Po kolejnym zatwierdzeniu (ścieżki) rozpoczyna się transfer plików do serwera.
W ten sposób zakończyliśmy aktualizacje sterowników Windows na platformę Intela.

4.9. Wydajność Samby

Wydajność samby:

Szybkość kopiowania różnych plików protokołem SMB z serwera samby i do niego, przewyższa rodowite serwery Microsoftu, Samba bardzo dobrze spisuje się przy większych ilościach stacji.
Istnieje wiele “trybików” które należy podkręcić, aby cała machina (samba) śmigała jak trzeba.
Znaczący wpływ na szybkość ma sprzęt, mianowicie: dysk twardy, procesor (magistrala), karta sieciowa, hub, liczba hostów w sieci. Wiemy dokładnie że przepustowość kart sieciowych o wielkości 10Mb/s to na te czasy muł, procesor serii MMX także archaizm minionej epoki, wreszcie dysk 3600RPM :(stosowanie hub'ów w sieci prowadzi do kolizji, tych wykrywanych i tych które nie zostały wykryte przez sprzęt. Należy unikać podobnych konfiguracji (jeśli budżet pozwoli).

Następnym etapem do poprawy wydajności naszej samby to kwestia ustawień programowych samej samby jak i całego systemu. Najważniejsze opcje – plik smb.conf:

log level = wartość powyżej 3 znacznie spowolni działanie serwera zasypując go mnóstwem danych o pracy serwera.

read raw = wartość logiczna umożliwia wykonywania szybkich strumieniowych odczytów przez TCP przy wykorzystaniu buforów 64KB, zalecana.

write raw = wartość logiczna umożliwia wykonywanie szybkich strumieniowych odczytów przez TCP przy wykorzystaniu buforów 64KB, także zalecana.

Blokady oportunistyczne to taka bestia która pozwala na zapis lokalny plików przez klienta, co podnosi do 30 wydajność serwera. Użycie:

fake oplocks = wartość logiczna zwolnienie z blokady plików które mogą być przetrzymywane przez innych klientów.

oplocks = wartość logiczna umożliwia lokalne buforowanie plików.

veto oplocks files = ścieżka nie zezwala na lokalne buforowanie plików podanych w ścieżce.

max xmit = wartość (2048 domyślna) określa maksymalny rozmiar pakietów, parametr służący do optymalizowania wolnych łącz.

read size = bajty (65 536 domyślna) określa opcję buforowania dla serwerów, w których prędkość dysków nie odpowiada prędkości sieci, wymaga eksperymentowania.

Okno zbiorcze TCP/IP
socket options = lista opcji gniazd ustawia opcje gniazd specyficzne dla systemu operacyjnego. Opcja "SOKEEPALIVE" powoduje że protokół TCP co cztery godziny sprawdza, czy klient jest wciąż dostępny. Opcja "TCPNODELAY" powoduje wysyłanie nawet niewielkich pakietów w celu zmniejszenia zwłoki. Obie zalecane. Inne opcje samby:

hide files = wzorzec podanie wzorca określającego pliki które powinny zostać ukryte, spowalnia serwer.

lpq chace time = czas ustawia się w momencie kiedy samba niepotrzebnie uruchamia kolejne zapytania w ramach kolejki druku – domyślna i rozsądna wartość to 10sec.

strict locking = wartość logiczna opcja mówi sambie aby sprawdzała przy każdym dostępie do pliku jego blokady, a nie wtedy kiedy klient tego wymaga – spowalnia serwer.

strict sync = wartość logiczna samba zapisuje każdy pakiet na dysku i czeka na dokończenie zapisu, kiedy klient ustawi bit synchronizacji w pakiecie. Windows 98 wysyła ten bit przy każdym połączeniu – w efekcie użytkownicy zauważą że serwer samby działa bardzo wolno.

sync always = wartość logiczna ustawienie tej opcji na YES każe sambie zapisywać fizycznie na dysku danych. Ma to sens w chwilach awarii – pozatym bardzo spowalnia prace serwera.

wide links = wartość logiczna uruchamia opcję podążania za dowiązaniami symbolicznymi w unixach, zalecana z opcja getwd cache, aby buforować pewną ilość danych. Istnieje także opcja follow symlink – którą można wyłączyć aby całkowicie zapobiec podążaniu za dowiązaniami.

ĆWICZENIE 5

Perl

5.1. Liczby

Liczb nie trzeba w żaden sposób oznaczać, mogą być przypisywane wprost do zmiennych.
Liczby zmiennoprzecinkowe

Kilka przykładów liczb zmiennoprzecinkowych:
0.1, -3.14, 2.71828,
Liczby całkowite (Integers)

Zmienne typu Integers to wszystkie liczby całkowite, włączając w to wartości dodatnie, ujemne i zero: ... -3, -2, -1, 0, 1, 2, 3 ....

Kilka przykładów:

12, -50, 20, 185, -6654, 6654

Pod spodem kilka złych przykładów które nie są liczbami całkowitymi:

15.5, -3.458, 3/2, 0.5

Inne systemy liczbowe

W perlu można używać wartości w systemach szesnastkowych, ósemkowych i binarnych. Dodatkowe informacje można znaleźć w Wikipedii:

* Zapis szesnastkowy
* System ósemkowy
* System binarny (dwójkowy)

Użycie liczb innych niż dziesiętne musi zostać zaznaczone. Liczby binarne oznacza się wpisując 0b, kilka przykładów:

0b101011101

0b10

Przy liczbach ósemkowych dodajemy na początku 0 (”zero”), kilka przykładów:

015462

062657

012

Wartości szesnastkowe zaznacza się poprzez 0x, przykłady::

0xF17A

0xFFFF

Operacje na liczbach

Operacje na liczbach są dość oczywiste, poniżej kilka prostych przykładów. +, - , /, i *

100 + 1 To 101

100 - 1 To 99

100 / 2 To 50

100 * 2 To 200

Teraz trochę mniej oczywiste operacje. Potęgowanie

W celu podniesienia liczby do potęgi używamy operatora **. Zobacz przykład:

5**2 To 25, jest to równoznaczne z 52

5.2. Zmienne

Zmienne pozwalają przechowywać potrzebne programowi dane. Każda zmienna ma swoją nazwę, dzięki czemu możemy łatwo się do niej odnosić. W Perlu użycie zmiennych wygląda następująco:
$liczba = 5;
print $liczba;
Uruchomiony program wypisze na ekranie:

5

Nazwa każdej zmiennej w Perlu musi być poprzedzona znakiem zaznaczającym jej typ. W przeciwieństwie do tzw. języków programowania o silnej typizacji dla Perla nie ma większego znaczenia, czy zmienna jest napisem, liczbą całkowitą czy ułamkiem. Rodzaje zmiennych są zupełnie inne.

W Perlu występuje tylko 5 typów zmiennych :

$skalarne, @tablice, %hasze, &podprogramy oraz *zmienne globalne.

W przykładzie możemy poznać, że $liczba to zmienna skalarna, gdyż jej nazwę poprzedza znak dolara. Spróbuj

Poeksperymentuj z przykładowym programem, podstawiając pod zmienną $liczba różne wartości (liczby, napisy itp.). Jak myślisz, co się stanie, gdy przed instrukcję print wstawimy linijkę $liczba = $liczba + 10?

Jeśli masz doświadczenie z tradycyjnymi językami programowania, zapewne zastanawiasz się, w jaki sposób w Perlu tworzy się zmienne.

Zapamiętaj:
Perl domyślnie tworzy zmienną w momencie, gdy po raz pierwszy na nią się natknie.

Fakt ten może być szokujący dla programistów, którzy mieli wcześniej doświadczenie z językami typu C czy Pascal, które ściśle przestrzegają wymogu zadeklarowania zmiennej przed jej użyciem. Perl jest językiem interpretowanym i jego interpreter sam zarządza pamięcią, w której działa program. W przeciwieństwie do języków kompilowanych, Perl nie jest zdezorientowany, gdy napotyka zmienną, o której do tej pory nie wiedział - w takim wypadku po prostu ją tworzy. Wymuszenie deklarowania zmiennych

Takie działanie może być potencjalną przyczyną błędów. Załóżmy, że w naszym programie popełniliśmy literówkę:

$liczba = 5; print $licba + 10;
W efekcie nasz program wydrukuje:

10

Nie tego oczekiwaliśmy, prawda? Przyczyną tego zachowania jest nasz błąd w pisowni nazwy zmiennej - kompilator nie znalazł zmiennej $licba więc stworzył nową, o wartości zero. Wymóg deklarowania zmiennych w językach kompilowanych typu C ma tę zaletę, że informuje o błędzie przy tego typu pomyłkach. W Perlu wygoda używania zmiennych może też być pułapką w przypadku większych programów. Możemy się jednak z tym uporać. Dodanie do programu linijki:

use strict 'vars';

Linijka ta mówi interpreterowi Perla, by odmawiał korzystania ze zmiennych, które nie były zadeklarowane. Po użyciu tej dyrektywy będziemy musieli w całym programie deklarować zmienne. Można to robić używając polecenia my:

my $liczba = 5;
Domyślnie zmienne w Perlu są globalne. my powoduje, że zmienna staje się lokalna. Więcej na ten temat wyjaśnione jest w następnych rozdziałach.

5.2.1. Podstawowe

Zwykłe zmienne

Zmienne skalarne określa się poprzez użycie $, mogą zawierać prawie dowolne typy danych. Na przykład:

$zmienna = 3; liczby całkowite
$zmienna = 3.1415926; zmiennoprzecinkowe
$zmienna = 3.4028236692093846346 33e+38; liczby w zapisie wykładniczym
$zmienna = $innazmienna + 1; wyliczane
$zmienna = 'Moge zawierac tekst'; tekstowe
$zmienna = $OdwolanieDoInnejZmiennej; referencje - odwołujące się do innej zmiennej
$zmienna = OdwolanieDoTablicy; referencje do tablic - odwołujące się do tablic

Wartość każdej z tych zmiennych wypiszemy poleceniem:

print $zmienna;

5.2.2. Tablice

Tablice

Tablice dla odróżnienia od innych zmiennych oznacza się znakiem @.

@tablica = (1,2,3,4,5,6,7,8,9,10); tablica liczb
@tablica = (1..10); zakres
@tablica = qw/1 2 3 4 5 6 7 8 9 10/; inny sposób
@tablica = split(/,/, "1,2,3,4,5,6,7,8,9,10"); tablica powstała z dzielenia napisu
@tablica = qw/czerwony niebieski zielony/; lista napisów
@tablica = qw/czerwony niebieski 1 zielony 5/; tablica mieszanego typu
@tablica = qw/tablica1 tablica2 tablica3/; tablica tablic

Poniższy fragment kodu wypisze wartości dowolnej z poprzednich tablic:

foreach $element (@tablica)
print "Nastepnym elementem jest $element"

W przypadku gdy działasz na pojedynczym elemencie tablicy (posługując się nawiasami kwadratowymi, aby uniknąć zamieszania), element ten jest uznawany za skalar i poprzedza się go znakiem $ tak jak przy innych zmiennych skalarnych.

$tablica[0] = 1;

Pierwszy element w tablicy ma numer 0 (chociaż to, jak i wiele innych rzeczy w Perlu może być zmienione). Jako indeks (numer elementu) tablicy może być użyta wartość zmiennej:
$tablica[$numer] = 1;

5.2.3. Tablice asocjacyjne

Tablice asocjacyjne

Tablice asocjacyjne (ang. associative arrays) nazywane w Perlu haszami (ang. hash) są oznaczane znakiem %. Hasz przechowuje pary klucz-wartość (ang. key-value), gdzie klucz i wartość mogą być zmiennymi dowolnego typu. Hasze możemy traktować jako tablice, które nie są numerowane, tylko jako indeksu używa się np. łańcuchów znakowych.

%hasz = ('klucz1' =¿ 'wartosc1', 'klucz2' =¿ 'wartosc2'); klucz i wartość zwyczajowo rozdzielamy strzałką
%hasz = ('jeden', 1, 'dziesiec', 10, 'sto', 100); ale można to tez robić przecinkiem

Jeśli używamy strzałki =¿ lewa strona powinna być umieszczona w cudzysłowach. Dla długich list jest dobrze umieszczać kolejne pary w osobnych liniach:

%hasz = (
'klucz1' =¿ 'wartosc1',
'klucz2' =¿ 'wartosc2',
'klucz3' =¿ 'wartosc3',
);

Do pojedynczego elementu odnosimy się używając nawiasów klamrowych, element ten jest wtedy uznawany za skalar i przyjmuje identyfikator $:

$hasz'klucz1' = 'wartosc1';

print $hasz'klucz1';

5.2.4. Podprogramy

Podprogramy

Podprogramy są zdefiniowane przez słowo kluczowe sub i mogą być wywoływane używając &, chociaż nie jest to zalecane. Poniższy podprogram liczy wartości ciągu Fibonacciego:

```
sub fib
my $n = shift;
return $n if $n ¡ 2;
return fib( $n - 1 ) + fib( $n - 2 );


print fib(14);
```

Szczegółowe omówienie wyżej wymienionych typów zmiennych pojawi się w podrozdziale Typy danych, natomiast podprogramów w podrozdziale Funkcje.

5.3. Operatory

5.3.1. Arytmetyczne

Wiele arytmetycznych operatorów jest binarnymi. Mają one dwa argumenty. Unarne operatory potrzebują tylko jednego argumentu.

Podstawowe operacje to: dodawanie (+), odejmowanie (-), mnożenie (*) i dzielenie (/).

Operator % (mod) oznacza resztę z dzielenia.

print 4 % 3; wynik 1 – 3 mieści się 1 raz w 4, reszta to 1.
print 4 % 2; wynik 0 – 4 / 2 = 2, czyli zero jest wynikiem - nie ma reszty.

Operatorem podnoszącym do potęgi jest **. Jeśli podnosimy do ułamka, dostajemy pierwiastek liczby. W poniższym przykładzie, jeśli drugi wynik podniesiemy do potęgi drugiej, powinniśmy zobaczyć wynik 2

(2 ** (1/2)) ** 2 = 2

print 4 ** 2; wynik 16 – 4 do potęgi 2.
print 2 ** (1/2); wynik 1.4142135623731 – pierwiastek z 2.

Operator auto-dekrementacji (–) jest operatorem unarnym. Zmniejsza on wartość liczby o jeden. Są dwa sposoby używania tego operatora.

$foo = 1;
print $foo–; post-dekrementacji – $foo jest pokazywany, a następnie jego wartość jest zmniejszana o 1.
print –$foo; pre-dekrementacji – $foo jest zmniejszany o 1, a potem pokazywany.

Operator auto-inkrementacji (++) jest operatorem unarnym. Inkrementuje (zwiększa) on wartość liczbową lub wartość łańcucha znaków. Są dwa sposoby używania tego operatora.

$foo = 1; print $foo++; post inkrementacja – $foo jest pokazywany, a następnie jego wartość jest zwiększana o 1.
print ++$foo; pre inkrementacja – $foo jest inkrementowany, a potem pokazywany.

$foo = 'a09'; print $foo++; post inkrementacja – $foo jest pokazywany, a następnie jego wartość jest zwiększana do 'a10'.
print ++$foo; pre inkrementacja – $foo jest inkrementowany do 'a11', a potem pokazywany.

$foo = 'a'; print $foo++; post inkrementacja – $foo jest pokazywany, a następnie jego wartość jest zwiększana do 'b'.
print ++$foo; pre inkrementacja – $foo jest inkrementowany do 'c', a potem pokazywany.

Gdy inkrementujesz łańcuchy znaków (litery i cyfry) możesz użyć operatora zakresu (print 'a' .. 'ac';). Prowadzi to czasem do nieoczekiwanych rezultatów.

5.3.2. Przypisywanie

Podstawowym operatorem przypisywania jest "=", który ustawia wartość po lewej stronie, aby była równa wartości po prawej. Zwraca także tę wartość. Tak więc może napisać $a = 5 + ($b = 6), który przypisze $b na wartość 6, a $a wartość 11 (5 + 6). Dlaczego potrzebowałbyś tak robić, jest innym zagadnieniem.

Perl obsługuje też operatory przypisania zapożyczone z C ("+=", "-=" itp.), ale ma także własne, niespotykane w C.

* "+=" (przypisanie dodające)

$a += $b;

Powyższy kod jest tym samym co "$a = $a + $b;". Reszta operatorów przypisania postępuje podobnie.

* "-=" (przypisanie odejmujące)
* "*=" (przypisanie mnożące)
* "/=" (przypisanie dzielące)
* "%=" (przypisanie modulo (reszta z dzielenia))
* "**=" (przypisanie potęgujące)
* ".=" (przypisanie dodające ciąg znaków)
* "x=" (przypisanie powtarzania)
* "&&=" (logiczne przypisanie AND)
* "——=" (logiczne przypisanie OR)
* "&=" (przypisanie bitowe AND)
* "—=" (przypisanie bitowe OR)
* "≙" (przypisanie bitowe XOR)
* " =" (przypisanie bitowe NOT)
* "«=" (przypisanie przesuwające bit w lewo)
* "»=" (przypisanie przesuwające bit w prawo)

5.3.3. Porównania

W perlu rozróżnia się porównywanie cyfr i znaków. Między innymi dlatego, iż w większości przypadków perl zamienia liczby na cyfry i odwrotnie, najczęściej jest to przydatne. Jednak niekoniecznie przy porównaniach.

Numery porównuje się:

* "==" (równa się)
* "!=" (nie równe)
* "¡" (mniejsze)
* "¿" (większe)
* "¡=" (mniejsze, bądź równe)
* "¿=" (większe, bądź równe)
* "¡=¿" (porównanie)

Operator "¡=¿" (porównanie), czasami nazywany "spaceship operator" (z powodu filmu "Gwiezdne wojny"), zwraca 0 jeśli obie wartości są równe, 1 jeśli lewa strona jest większa lub -1 jeśli prawa jest większa.

Łańcuchy znaków porównuje się:

* "eq" (równa się)
* "ne" (nie równa się)
* "lt" (mniejsze)
* "gt" (większe)
* "le" (mniejsze bądź równe)

* "ge" (większe bądź równe)
* "cmp" (porównanie).

5.3.4. Logiczne

Operatory logiczne są używane w porównaniach. Zwracają one wartości logiczne (tj. prawdę lub fałsz). Wyglądają one następująco:

"&&" (logiczne AND, zwraca prawdę gdy lewa i prawa strona jest prawdą)

prawda jeżeli 1 jest równe 1 i zmienna $a jest większa od trzech
$a = 4;
print 1 == 1 && $a ¿ 3; wyświetli 1

"——" (logiczne OR, zwraca prawdę gdy lewa lub prawa strona jest prawdą)

prawda jeżeli 2 jest większe od 1 lub jeżeli 4 jest większe od 10
print 2 ¿ 1 —— 4 ¿ 10; wyświetli 1

"!" (logiczne NOT, zwraca prawdę gdy wyrażenie jest fałszywe)

prawda jeżeli 2 == 3 jest fałszem
print !(2 == 3); wyświetli 1

Operator "!" ma wyższy priorytet, niż operator porównania dlatego konieczne było użycie nawiasów.

Można także używać operatorów łatwiejszych w czytaniu takich jak "and" (odpowiada &&), "or" (odpowiada ——) oraz "not" (odpowiada !). Trzeba tylko pamiętać, że mają one niższy priorytet.

print 1 == 1 and 2 ¿ 3; program wyświetli 1 i nie przejdzie do nowej linii...

Stało się tak dlatego, że porównanie miało inną postać:

print (1 == 1) and 2 ¿ 3;

Poprawiona wersja wygląda następująco:

print (1 == 1 and 2 ¿ 3,);

5.3.5. Ocenianie skrótowe

Perl używa oceniania skrótowego zwanego także "ocenianiem leniwym". Gdy dasz do porównania perlowi:

1 ¿ 2 && 3 == 3;

to po ocenieniu, że pierwsza część nie ma sensu ("1 ¿ 2" zwróci fałsz) nie będzie on sprawdzał następnego warunku, ponieważ i tak całe wyrażenie ma już wartość 0.

5.4. Pętle

5.4.1. if

Instrukcja if jest najważniejszą strukturą warunkową w Perlu. Składnia jest taka jak pokazano niżej:

if (wyrażenie logiczne)
instrukcje

Jeśli wyrażenie logiczne jest prawdziwe, instrukcje zawarte między nawiasami klamrowymi zostaną wykonane. Wyrażenie logiczne może zawierać dowolne operatory porównania opisane wcześniej. Może także zawierać operatory logiczne. Na przykład:

if (($x ¡ -10) —— (($x ¿ 0)&&($x ¡ 10)))
print "x jest mniejsze niż -10 lub jest ściśle zawarte między 0 a 10;

Instrukcje warunkowe mogą także zostać rozszerzone poprzez struktury elsif i else:

if (wyrażenie logiczne 1)
instrukcje 1
elsif (wyrażenie logiczne 2)
instrukcje 2
else
instrukcje 3

Działa to następująco. Program po kolei sprawdza wszystkie wyrażenia logiczne. Jeżeli któreś z nich zwróci wartość prawdziwą zostaną wykonane instrukcje zawarte w odpowiednim bloku po czym program wyjdzie z instrukcji warunkowej if. Jeżeli wszystkie wyrażenia zwrócą wartość fałszywą zostanie wykonany blok else.

$x = 5;
if ($x ¿ 10)

print "$x jest większe od 10;
elsif ($x ¿ 2)
print "$x jest większe od 2;
elsif ($x == 5)
print "$x jest równe 5";
else
print 'Nie wiem ile wynosi $x';

Powyższy program wyświetli napis "5 jest większe od 2" i zakończy działanie. Dzieje się tak dlatego, że perla nie obchodzi czy wyrażenia dalej też spełniają warunek.

Uwaga! W przeciwieństwie np. do języka C w Perlu instrukcja nie może występować bezpośrednio po warunku - musi znajdować się wewnątrz nawiasów klamrowych. Na przykład poniższy kod zostanie uznany za nieprawidłowy:

if ($x ¡ -10) print "x jest mniejsze niż -10"; blad!

Można też użyć zamiast if słowa kluczowego unless - znaczy tyle, co if z zaprzeczeniem. Tak więc kod

unless (wyrażenie logiczne)
instrukcja

jest równoważny

if (!(wyrażenie logiczne))
instrukcja

Po unless można oczywiście używać elsif i else tak jak w normalnym if.

5.4.2. while

While i until

Składnia pętli wygląda następująco:

while (wyrażenie logiczne)
instrukcje

Instrukcje zawarte w pętli while są wykonywane, dopóki wyrażenie logiczne jest prawdziwe. Czyli, jeśli na samym początku wyrażenie to nie jest prawdziwe, to instrukcje zawarte w pętli nie zostaną wykonane ani razu.

Pętla until neguje sens wyrażenia logicznego. Jest wykonywana dotąd dopóki wyrażenie jest fałszywe.

Pętle while i until jeżeli tego zechcemy mogą posiadać blok continue:

while (wyrażenie logiczne)
instrukcje
continue
instrukcje

Instrukcje zawarte w tym bloku są wykonywane po skończeniu bloku pierwszego lub po użyciu operatora next omówionego w dalszej części tego rozdziału.

Oto prosta pętla wypisująca liczby od 0 do 100:

my $n = 0; while ($n ¡= 100)
print $n++;

Pętle do while i do until są technicznie modyfikatorami i tak naprawdę nie są strukturami sterującymi.

do
instrukcja
while (wyrażenie logiczne);

do
instrukcja
until (wyrażenie logiczne);

5.4.3. for

Słowa kluczowe for i foreach są synonimami i mogą być używane zamienne. Pamiętaj jednak, że w dobrym stylu programowania jest aby używać słowa foreach dla pętli pobierającej wartości z listy, zaś for dla pętli zawierających warunki.

foreach zmienna (tablica)
instrukcje

Pętla for wykonuje instrukcje dla każdej zmiennej w tablicy. Aktualny element jest przechowywany w zmiennej. zmienna jest aliasem do każdego elementu w tablicy, idzie kolejno od pierwszego elementu poprzez pętlę. Pętla jest zakończana, jeśli każdy element został uwzględniony. Ponieważ zmienna jest aliasem, zmieniając jego wartość będziemy

zmieniać zawartość tablicy. Nie wolno tego robić, gdyż utrudnia to wychwytywanie błędów. Jeśli zmienna nie zostanie wpisana w kodzie programu, domyślnie zostanie użyta zmienna $_.

```
@tablica = ('dziwnie', 'wypisany', 'napis');
foreach (@tablica)
print "$_";
```

Przepisz kod powyżej i zobacz jak on działa.

Istnieją dwa sposoby użycia pętli for/foreach. Pierwszy już poznaliśmy, oto następny.

```
for (inicjacja; warunek zakończenia; wyrażenie inkrementujące)
instrukcje
```

Składa się ona z trzech wyrażeń rozdzielonych średnikami. Wszystkie są opcjonalne (oprócz średników), a domyślną wartością pętli jest prawda (więc bez wyrażeń pętla jest nieskończona).

Oto kolejna prosta pętla wypisująca liczby od 0 do 10:

```
for ($n = 0; $n ¡= 10; $n++)
print "$n;
```

W dobrym stylu jest deklarowanie zmiennych w pętli, stają się one wówczas zmiennymi lokalnymi. Unikamy też w ten sposób przypadkowego nadpisania zmiennych o takich samych nazwach dostępnych wewnątrz funkcji, lub w cały programie. Uważaj na wcześniejsze niż obecna wersje perla pod Windows, taka konstrukcja była traktowana jako błąd. Obecnie nie ma tego problemu. Przykłady:

```
for (my $n = 0; $n ¡= 10; $n++)
print "$n";
```

```
foreach my $n (@tablica)
print "$n";
```

5.4.4. Operatory sterujące pętlą

5.4.4.1. next

next powoduje przejście do następnej iteracji, nie zakończając aktualnej. Można go używać razem z etykietą albo bez niej. Jeżeli pętla zawiera blok continue to jest on

wykonywany.

```
my @tablica = ('dziwnie', 'wypisany', 2, 'napis');
for (@tablica)
if (/+/)
next;

print "$_";
continue
print "po kolejnej iteracji;
```

/+/ jest wyrażeniem regularnym, zwraca prawdę gdy odnajdzie cyfrę. Ale o tym dalej w podręczniku.
Przepisz ten program i uruchom. Teraz powinieneś już rozumieć operator next.

5.4.4.2. last

last powoduje opuszczenie bieżącej pętli. Blok continue nie jest wykonywany.

```
my @linie = ("jakiś tekst", "następna linia", "KONIEC", "inny fragment");
my $ile = 0;
for (@linie)
if (/^KONIEC/)
print "To już jest koniec...";
last;

print;
continue
print 'Zostałem wyświetlony po raz ', ++$ile;
```

Pętla pobiera elementy z tablicy i wyświetla je dopóki nie natrafi na napis "KONIEC" na początku lini (tak to też jest wyrażenie regularne).
Zastanów się co wyświetli ten program, a następnie uruchom go.

5.4.4.3. redo

redo ponownie wykonuje tę samą pętlę, nie sprawdzając warunku. Nie wykonuje bloku continue.

5.4.4.4. goto

goto powoduje natychmiastowy przeskok w inne wskazane miejsce w kodzie. np:

```
goto trzy;
jeden:
print "world";
goto koniec;
dwa:
print "hello ";
goto jeden; trzy:
print "napis ";
goto dwa;
koniec:
```

Powyższy program wyświetli tekst: "napis hello world". Jak łatwo się domyślić jest dużo znacznie łatwiejszych sposobów na wykonanie tego zadania.

Używanie tej komendy nie jest zalecane ze względu na duże gmatwanie kodu oraz łatwą zastępowalność innymi strukturami programistycznymi.

5.5. Funkcje

5.5.1. Wstęp

Na razie pisaliśmy tylko kilka linii w Perlu na raz. Nasz przykładowy program zaczynał się na początku pliku i rozrastał się do końca pliku, z kilkoma skokami używającymi słów kluczowych sterujących wykonaniem, jak if, else i while. W wielu przypadkach jest jednak użyteczne dodanie dodatkowej warstwy do organizacji naszych programów.

Na przykład, kiedy coś idzie źle, typowy program wypisuje komunikat o błędzie. Kod wypisujący komunikat o błędzie może wyglądać następująco:

```
print STDOUT "cos poszlo zle!";
```

Inne komunikaty mogą wyglądać trochę inaczej, na przykład

```
print STDOUT "cos INNEGO poszlo zle!";
```

Możemy ozdobić nasz kod setkami linijek takich jak powyższe i wszystko będzie działało dobrze... przez pewien czas. Ale wcześniej czy później będziemy chcieli oddzielić komunikaty o błędach od "komunikatów stanu", które przekazują niegroźne informacje o naszym programie. Aby to uczynić, możemy poprzedzić wszystkie komunikaty o błędach słowem "BLAD" i wszystkie komunikaty o stanie słowem "STAN". Kod typowego komunikatu o błędzie zmieni się na

```
print STDOUT "BLAD: cos poszlo zle!";
```

Problem jest taki, że mając setki komunikatów o błędzie nie jest łatwo zmienić je wszystkie. Jest to sytuacja, kiedy podprogramy mogą pomóc.

Wikipedia definiuje podprogram jako "sekwencję instrukcji, która przeprowadza specyficzne zadanie jako część większego programu. Podprogramy mogą być wywoływane z różnych miejsc programu, w ten sposób pozwalając programom na dostęp do podprogramu wiele razy bez przepisywania kodu podprogramu więcej niż raz".

Cała ta kwiecista mowa oznacza, że możemy zebrać kod komunikatu o błędzie w jednym miejscu, jak poniżej:

```
sub wypisz_komunikat_bledu
my($komunikat) = @_;
print STDOUT "BLAD: " . $komunikat ;
```

Kiedy tylko coś pójdzie źle w naszym programie, możemy aktywować, lub wywołać (call), ten podprogram, z takim komunikatem, jaki tylko chcemy:

```
...
wypisz_komunikat_bledu("zdarzylo sie cos zlego");
...
wypisz_komunikat_bledu("zdarzylo sie cos naprawde strasznego");
...
wypisz_komunikat_bledu("zdarzylo sie cos denerwujacego");
...
```

Zobaczymy komunikaty typu

BLAD: zdarzylo sie cos zlego

Jeśli chcemy zmienić format naszych komunikatów o błędzie, aby, powiedzmy, zawierały kilka wykrzykników, wystarczy po prostu zmienić podprogram:

```
sub wypisz_komunikat_bledu
my($komunikat) = @_;
print STDOUT "BLAD: " . $komunikat . "!!!";
```

Jest to na pewno prosty przykład i podprogramy mają kilka innych zalet, ale jest to skrótowy opis. Zasada jest jasna: napisz w jednym miejscu, popraw w jednym miejscu, zmień w jednym miejscu.

Teraz parę dalszych szczegółów na temat podprogramów. Będziemy dalej często używać następującego podprogramu, który, jeśli nie jest to jeszcze oczywiste, dodaje dwie liczby i zwraca ich sumę.

```
sub dodaj_dwie_liczby
```

```
my($x,$y) = @_;
my $suma = $x + $y;
return $suma;
```

5.5.2. Części podprogramu

Oficjalna składnia definiowania podprogramu jest następująca:

```
sub NAZWA PROTOTYP ATRYBUTY BLOK
```

Jeśli ma to dla ciebie jakikolwiek sens, prawdopodobnie nie musisz czytać tego podręcznika. Nazwa

```
sub dodaj_dwie_liczby
my($x,$y) = @_;
my $suma = $x + $y;
return $suma;
```

Pierwsza linia funkcji zaczyna się od słowa kluczowego sub (od subroutine czyli po angielsku podprogram), po którym następuje nazwa funkcji. Każdy ciąg liter (bez polskich znaków) i liczb, który nie jest zarezerwowanym słowem Perla (jak for, while, if czy else) jest poprawną nazwą funkcji. Podprogramy, których nazwa określa, co robi dany podprogram, czynią kod programu łatwiejszym w czytaniu.

5.5.3. Prototyp

```
sub dodaj_dwie_liczby($$)
```

Opcjonalne ($$) określa, ilu argumentów oczekuje podprogram. ($$) mówi "ta funkcja wymaga dwóch wartości skalarnych". Z powodu złożoności implementacji Perla, prototypy nigdy nie osiągnęły pełnego potencjału i najlepiej je pomijać.

5.5.4. Ciało

Ciało podprogramu wykonuje "pracę" i zawiera trzy podstawowe sekcje.

5.5.4.1. Czytanie argumentów

Dane przekazywane do podprogramu są nazywane argumentami lub parametrami rzeczywistymi. Na przykład, w

```
dodaj_dwie_liczby(3,4)
```

3 i 4 są argumentami podprogramu dodaj_dwie_liczby.

Perl przekazuje argumenty do podprogramu jako tablicę reprezentowaną przez @_. Zazwyczaj jest wygodniej nadać znaczące nazwy tym argumentom, więc pierwsza linia funkcji wygląda najczęściej następująco:

```
sub dodaj_dwie_liczby
my($x,$y) = @_; czytanie argumentow
my $suma = $x + $y;
return $suma;
```

co umieszcza zawartość @_ w dwóch zmiennych nazwanych $x i $y. $x i $y są nazywane parametrami formalnymi. To rozróżnienie między parametrami formalnymi (argumentami) i parametrami rzeczywistymi jest subtelne i w większości nieistotne. Uważaj, aby nie pomylić specjalnej zmiennej $_ z @_, czyli tablicą argumentów przekazywanych do funkcji.

Pewne podprogramy nie wymagają żadnych argumentów, na przykład

```
sub hello_world
print STDOUT "Hello World!";
```

wypisze "Hello World" na STDOUT. Ten podprogram nie wymaga żadnych dodatkowych informacji o tym, jak wykonać swoją pracę i z tego powodu nie potrzebuje żadnych argumentów.

Większość nowoczesnych języków programowania ułatwia programiście życie jawnie dzieląc listę argumentów na zmienne. Niestety, Perl tego nie robi.

W programowaniu parametr oznacza niemalże to samo, co argument. Obydwu pojęć można używać zamiennie.
Ważna uwaga: zmienne globalne i lokalne

W odróżnieniu od języków programowania takich jak C lub Java, wszystkie zmienne stworzone lub użyte w podprogramach Perla są domyślnie zmiennymi globalnymi. Oznacza to, że każda część programu poza twoim podprogramem może modyfikować te zmienne oraz że twój podprogram może, nie wiedząc o tym, modyfikować zmienne, których nie ma powodu zmieniać. W małych programach jest to często wygodne, ale gdy program staje się większy, prowadzi to często do złożoności i jest uważane za kiepski zwyczaj.

Najlepszym sposobem na uniknięcie tej pułapki jest umieszczenie słowa kluczowego my przed wszystkimi zmiennymi gdy pojawiają się po raz pierwszy. Mówi to Perlowi, że chesz, aby te zmienne były dostępne tylko wewnątrz najbliższej grupy nawiasów klamrowych zawierającej tę instrukcję. W efekcie te lokalne zmienne działają jako notatnik używany wewnątrz podprogramu, który znika, gdy podprogram kończy działanie. Linia

use strict; na początku programu poinstruuje Perla, aby wymusił użycie my przed każdą zmienną, by uniemożliwić ci przypadkowe stworzenie zmiennych globalnych.

Alternatywą do my, którą możesz spotkać w starszych programach Perla jest słowo kluczowe local. local jest w pewnym stopniu podobne do my, ale trudniej jest go używać. Najlepiej przyzwyczaić się do my w swoich programach.

"Zasięg" opisuje, czy zmienna jest lokalna czy globalna i parę innych zależności. Informacji o nim możesz poszukać w Wikipedii lub innych źródłach.

5.5.4.2. Wyrażenie return

Ostatecznie niektóre podprogramy "zwracają" (ang. "return") część przetworzonych informacji, używając słowa kluczowego return.

```
sub dodaj_dwie_liczby
my($x,$y) = @_;
my $sum = $x + $y;
return $sum; zwracanie wartości
```

Na przykład wyrażenie:

$sum = dodaj_dwie_liczby(4,5);

umieści w zmiennej $sum wartość 9, czyli sumę 4 i 5.

return może być także użyty bez żadnych argumentów, by opuścić podprogram zanim parser dotrze do końcowego

```
sub hello
print "To zostanie wydrukowane";
return;
print "A to nie";
```

5.5.5. Odwołanie do programów

Podprogramy mogą być zdeklarowane gdziekolwiek w kodzie programu. Wywoływane mogą być korzystając ze składni:

dodaj_dwie_liczby(4,5); najbezpieczniejsze podejście

dodaj_dwie_liczby 4, 5; tylko jeżeli predeklarowana

&dodaj_dwie_liczby(4,5); stara składnia Perla, ale nadal ważna

Jeżeli prefiks & nie jest użyty branie w nawias jest wymagane, chyba że podprogram był predeklarowany

5.5.6. Funkcje wywołujące funkcje

Funkcje są bardzo ważnym krokiem do efektywnego kodu, lecz dopiero łączenie funkcji wyzwala ich potęgę.

Jak być może podejrzewasz, wywoływanie funkcji wewnątrz innej nie różni się w żaden sposób od wywołania jej z części programu, która znajduje się poza wszelkimi "klamerkami".

Ta funkcja dodaje dwie liczby i mnoży je przez 3. Musisz ścierpieć jej bezużyteczność, kiedy będziesz budował swoje programy by rozwiązać unikalne problemy, natychmiast odkryjesz "moc" tego stylu programowania.

```
sub dodaj_dwie_liczby_i_pomnoz_przez_trzy  my($x,$y) = @_; wczytaj parametry
my $sum = dodaj_dwie_liczby($x,$y); dodaj x i y, a wynik umieść w $sum
my $sum_razy_trzy = $sum * 3; pomnóż przez trzy
return $sum_razy_trzy; zwróć wynik
```

Linia

```
my $sum = dodaj_dwie_liczby($x,$y); dodaj x i y, a wynik umieść w $sum
```

wywołuje naszą funkcję odpowiedzialną za dodanie dwóch liczb i zwraca wynik do naszej zmiennej $sum. Rzecz to prosta...

W tej funkcji napisaliśmy za dużo kodu, niż w rzeczywistości potrzeba. Może być to mniejsze, choć równie czytelne:

```
sub dodaj_dwie_liczby_i_pomnoz_przez_trzy
my($x,$y) = @_; wczytaj parametry
return 3 * dodaj_dwie_liczby($x,$y); zwróć dodane x i y, pomnożone przez 3
```

5.5.7. Rekursywne wywoływanie funkcji

Zobaczyliśmy funkcje wywołujące inne funkcje, lecz udanym pomysłem w programowaniu są funkcje wywołujące siebie. To nazywa się rekursją. Na pierwszy rzut oka następstwem rekursji może być przyczyną rzekomej nieskończonej pętli,lecz jest to z całą pewnością standard programowania.

W matematyce silnią nazywamy iloczyn wszystkich dodatnich liczb naturalnych nie większych niż dana liczba. Dla przykładu silnia z 5 (zazwyczaj pisana jako 5!) jest obliczana pomnożeniem przez siebie liczb 5, 4, 3, 2 i 1. Oczywiście jedynka nie zmienia

wyniku. Funkcja silni może być przydatna do obliczania rzeczy typu liczby możliwych przypadków rozmieszczenia naszych gości przy stole.

Silnia jest oczywistym przykładem funkcji, która rekursywnie może być napisana równie prosto jak za pomocą pętli.

```
sub silnia
my($num) = @_;
if($num == 1)
return 1; zatrzymaj przy 1, nie mnóż przez zero
else
return $num * silnia($num - 1); wywołaj funkcje silni rekursywnie
```

Samowywołująca siebie linia to:

```
return $num * silnia($num - 1);
```

która wywołuje funkcję silni z funkcji silni. Mogło by się tak dziać zawsze, gdybyśmy nie mieli swego rodzaju "znaku stopu", który temu zapobiega:

```
if($num == 1)
return 1;
```

To zatrzymuje sekwencje wywołań do silni i zapobiega nigdy niekończącej się pętli.

Działanie takiej funkcji można ująć w taki sposób:

```
silnia(5)
= 5 * silnia(4)
= 5 * 4 * silnia(3)
= 5 * 4 * 3 * silnia(2)
= 5 * 4 * 3 * 2 * silnia(1)
= 5 * 4 * 3 * 2 * 1
= 120
```

Zaledwie jednak musnęliśmy rekursję. Dla niektórych problemów programistycznych jest to naturalne rozwiązanie. Dla innych jest to trochę... dziwne rozwiązanie. Wystarczy powiedzieć, że to narzędzie, które każdy programista powinien mieć w zanadrzu.

5.5.8. Podprogramy, funkcje, procedury

Czytając tą lekturę mogłeś spotkać się z terminami "podprogramy", "funkcje", czy "procedury". Większość z nich była używana zamiennie, lecz ich docelowe znaczenie jest

następujące:

Funkcja zawsze zwraca wartość. Pełno funkcji możemy znaleźć w takich klasach jak math.

Procedura może nie zwracać wartości. W odróżnieniu od funkcji, procedura zwykle oddziałuje z zewnętrznym środowiskiem jej argumentem. Na przykład, procedura może czytać i zapisywać do plików.

Podprogram wiąże się z sekwencją instrukcji, które mogą być albo funkcjami, albo procedurami.

Bibliografia

[1] Interbase *Data definition guide* DateDef.pdf

[2] Interbase *Language Reference* langref.pdf

[3] Vinkenoog P. *Firebird 2.1 Language Reference Update.*