update: 18.05.09

MAN BY DePe

OTOCZENIE

Moje środowisko testowe na platformie Windows XP:

1. Sun VirtualBox 2.2.2
   - ustawienie w grupie NET: Bridged Adapter
   - instalacja Debian 5.0.1 Lenny (NETINST)
2. PuTTY 0.60
   - warto ustawić Window-Translation: UTF-8
   - wcześniej #apt-get install ssh
3. WinSCP3 ver 3.8.2

WINDOWS XP – 192.168.2.233
GNU/Linux DEBIAN – 192.168.2.234
GATEWAY: 192.168.2.1

ROZRUCH:

Kalkulator, wpisujesz np 2+2 <Enter> ;-)
#bc

Kalendarz, kiedy wypada wielkanoc w 2009 ?
#ncal –e 2009
#ncal –eo 2009 (w/g Prawosławnych)
#ncal kwiecień 2009

Skąd wiadomo jakie polecenia, hm.... np. tak:
#apropos calendar

#man ncal (i gotowe!)

Filtrowanie – to polecenie wyświetli tylko wtorek (cały wiersz)

#ncal |grep wt

To polecenie wytnie słowo „dwa” (separatorem –d jest przecinek i pole nr -f)

#echo 'raz, dwa, trzy,' |cut –d, -f2 (wypisze: dwa)

#echo 'raz, dwa, trzy,' |cut –d, -f1,3 (wypisze: raz trzy)

#ncal > kalendarz.txt (zapisanie do pliku tekstowego wyniku operacji)

#ncal |sort |uniq (najpierw posortowanie i wypisanie tylko tych linijek które są unikatowe, uniq porównuje linijki obok siebie dlatego najpierw sort)

#grep cosik (przepuszcza tylko linijkę zawierającą cosik)

#eject (wysuwa tackę napędu CD, dobre do diagnozy w LAN)

#false (program nic nie robi zwraca false)

#yes (wyrzuca cały czas yes np.: yes |rm *)

#uname –a (pokazuje wersję kernela i nazwę)

POMOC:

#man passwd (pokazuje tylko opis polecenia)

#man 5 passwd (sekcja 5 dotyczy plików konfiguracyjnych

#man –a passwd (pokazuje wszystkie sekcje ALL)

#whereis passwd (szuka w całym systemie)

#updatedb (odświerza indeksy dla polecenia LOCATE)

#locate ncal (szybkie szukanie położenia pliku)

#apropos pci (szuka w manualu)

#which lspci (zwraca /usr/bin/lspci)

#echo $PATH (wyświetla ścieżki poszukiwań)

#stat `which passwd` (pokazuje dokładne dane pliku)

#find <gdzie> <filtry>

#find /bin –name ls

#find / -user kurs (wszystkie pliki gdzie właścicielem jest kurs)

#find /sbin –uid 1000

#find / -type f –perm /o+x (znajdzie wszystkie pliki wykonywalne przez pozostałych)

#find / -type s (znajdzie wszystkie gniazda)

#find / -perm -1000 (stickibit)

#find / -perm -4000 (setuid)

#find / -perm /u+s -exec stat {} \; (szuka wszystkich z setuid i wykona na tych znalezionych stat)

#find / -type f |wc –l (pokazuje ile jest wszystkich plików na dysku)

ŚLEDZENIE LOGÓW:

#tail –f /var/log/messages (na osobnej konsoli mamy cały czas podgląd)

Sprawdzenie co się zmieniło np. po załadowaniu modułu

#lsmod >przed.txt

#lsmod >po.txt

#diff przed.txt po.txt –c (pokazuje różnice między plikami)

SCREEN:

#apt-get install screen

#screen (uruchomienie screena)

<ctrl a> c (tworzenie nowego screena)

<ctrl a> 0 <ctrl a> 1 itd. (przełączanie między screenami)

<ctrl a> d (usuwanie sesji)

<ctrl a> ? (pomoc kontekstowa)

STRUKTURA KATALOGÓW:

/ - korzeń drzewa katalogów, tzw ROOT

/proc - jądro zakłada ten katalog jest to odbicie wewnętrzne struktury jądra systemu (w linuxie wszystko jest plikiem ☺, można wykorzystać do komunikacji z jądrem np: cat /proc/sys/net/ipv4/ip_forward (0-wył, 1-wł forwardowanie pakietów), znajdziemy też plik kcore który zawiera zrzut pamięci RAM oraz cpuinfo który zawiera dane techniczne procesora. Katalog o nazwie 1 przypisany jest do INIT, np. tree i lspci pobierają info z tego miejsca

cat /proc/partitions plik pokazuje wszystkie partycje zamontowane i nie zamontowane

/dev – katalog reprezentujący urządzenia we/wy np. Wpisanie danych do pliku /dev/dsp spowoduje odtworzenie ich w głośniku

/var - to co ma się z założenia zmieniać (wszelkie logi)

/var/lib – tu najczęsciej programy zapisują swój stan np bazy danych tu się lokują

/opt – tu np wgrywa się JAVA zamiast standardowo do /bin i /lib

/usr - to co nie należy do rdzenia systemu (/bin, /sbin, /lib) oraz /share –dane programu które się nie zmieniają np logo, ikony itp

/usr/share/doc – dokumentacje

/usr/share/man – dokumentacje w formacie manuala MAN

/usr/share/info – podobnie jak MAN do podręczników systemowych

/usr/include – pliki nagłówkowe w formacie preprocesora C, umożliwiające kompilację

/usr/local – w Debianie domyślny katalog do instalacji (zawiera lib,bin,share,doc, include,sbin, etc)

/usr/src – tu są składane źródła

/bin – skompilowane programy uruchomieniowe

/sbin - binarki przeznaczone dla administratora lub systemu

/lib – biblioteki współdzielone (zestawy funkcji)

/lib/modules/`uname –r` (tu znajdziemy moduły ładowalne *.ko uszeregowane w podkatalogach, np. crypto –moduły kryptograficzne, driver -sterowniki)

/mnt – zwyczajowo do podmontowywania udziałów sieciowych np /mnt/windows

/media – tu są specjalne foldery w których możemy przeglądać zawartość np. cdrom, floppy, pendrive itp

/tmp - tymczasowe (ten katalog ma dodatkowe prawo t – sticki bit

/root – katalog domowy root'a

/home – katalogi domowe userów

/etc - 99,9% plików konfiguracyjnych

/etc/<nazwa>.d - katalog konfiguracyjny (zawiera zbiór małych plików konfiguracyjnych)

/boot – katalog dla GRUBa lub LILO (do bootowania), np. vmlinuz – to obraz jądra

/lost+found - miejsce gdzie są pliki odzyskane np. po działaniu fsck

/srv - katalog dla usług dodatkowych, np. dla potrzeb Apache do przechowywania stron hostów virtualnych

OZNACZANIE KATALOGÓW

. – oznacza aktualny katalog

.. – katalog nadrzędny

~ - katalog domowy aktualnego usera

/ - początek drzewa

OZNACZANIE PLIKÓW (ls –l pierwsza kolumna)

- zwykły plik (.plik – oznacza plik ukryty)

d katalog

l link symboliczny

p pipe (rureczka)

c urządzenie znakowe np. myszka, klawiatura

b urządzenie blokowe np. dysk

s socket (brama do komunikacji pomiędzy programami lokalnie, port –wymiana między programami w sieci)

BUDOWA PASSWD:

nazwa użytkownika: (login)

x: (oznacza że hasło jest w /etc/shadow)

UID: (numer usera)

GID: (numer grupy)

GECOS: (pełna nazwa usera, na ogół puste)

home_directory: (katalog domowy)

shell: (powłoka czyli to co po zalogowaniu)

(cat /etc/default/useradd tu definiujemy domyślną powłokę SHELL dla nowo tworzonych userów - plik szablonu)

BUDOWA SHADOW:

nazwa użytkownika: (login)

MD5 hasła: (hasło zakodowane plus salt w postaci godziny zmiany i numeru procesu)

Data kiedy było zmieniane:

Co ile ma być zmieniane hasło:

Przez ile dni będą ostrzeżenia:

Ile dni po którym hasło jest blokowane

(tym można zarządzać przez #chage)

#chage –l user (wyświetla dane konta user)

EDYTOR VIM

Protoplasta to: vi

Polecam też edytor: nano

<ESC> przejście do linii komend

:q wyjście

:q! wyjście bezwzględne bez zapisu zmian

:w zapis pliku (jeśli nie było nazwy to piszemy :w plik.txt)

:i przejście do trybu edycji

:v przejście do trybu virtualnego (zaznaczanie)

:y kopiuj

:x wytnij

:p wklej

u cofa zmiany

dd kasuje całą linię

3G skok do lini 3

:r <nazwa_pliku> (wczyta zawartość pliku w miejsce kursora)

:sh (wyjście do shella, szczególnie niebezpieczne w sudo dla zwykłego usera bo vim uruchamia się z prawem roota, powrót to: exit)

/ szukany_ciąg_znaków (np. / Jan.*Kowalski zapis .* oznacza dowolny znak występujący dowolną ilość razy)

MANGER MC:

#apt-cache search midnight

lub #apt-cache search commander

#apt-get install mc

F10 lub <ESC>0 wyjście z programu

<TAB> przeskok między oknami

<ESC> <Enter> przepisanie zaznaczonej nazwy do powłoki

<ESC>s wyszukiwarka

<CTRL> o pokazuje to co podspodem

ustawienie VIM zamiast NANO (jako domyślnego edytora w mc F4)

#export EDITOR=`which vim`

POWŁOKA BASH:

/bin/sh stanowi symboliczne dowiazanie do /bin/bash

Strzałki góra dół (przesuwa w historii)

Alt . (przywołanie ostatniego parametru)

Ctrl R <tekst> (cofnie nas do ostatniej pozycji w historii zawierającej tekst)

#history (wyświetla historię poleceń)

cat alamakota 2>nic

(w oknie terminala nic się nie wypisze a w pliku ‘nic’ komunikat o braku takiego pliku)

ls >plik 2>nic

0 standardowe WEJ

1 standardowe WYJ

2 standardowe WYJ Błędów

2>&1 (przekierowanie standardowego wyj błędów do standardowego wyj)

| potok, wiąże standardowe wyjście jednego procesu ze standardowym wejściem drugiego

sort <plik jest równoznaczne z zapisem **cat plik |sort**

przekierowanie wyniku do edycji od razu musi być vim - :

#ls –l |vim –

#tee plik (kopiuje standardowe wejscie na standardowe wyjscie zapisując przy okazji wszystko do pliku)

ls /etc |tee plik.txt

wc -word count

#ls |wc (oblicza wszystkie trzy parametry)

#ls |wc –l (-l liczba linii, -w ilość słów, -c ilość znaków)

np**. #cat passwd |wc –l** (oblicza użytkowników systemu)

ls |head –n 12 (wyrzuca 12 pierwszych linii)

ls |tail –n 12 (wyrzuca 12 ostatnich linii)

tr lancuch1 lancuch2 (każdy napotkany znak z pierwszego łańcucha zamienia na odpowiadający mu znak z drugiego łańcucha)

#echo „napis jakis tam" | tr a z (zamieni każde a na z)

#tr abc xyz (zamienia a na x, b na y, c na z)

#echo „Polskie ąęćżźół" | iconv –f utf-8 –t ASCII (konwerter znaków)

#date (pokazuje aktualną datę i czas systemowy)

#date +%Y.%m.%d (zwraca datę w konkretnym formacie 2009.04.30)

#date +%A --date='2009-04-30' (zwroci dzień tygodnia np. czwartek)

echo 'uname –r' (to wypisze napis uname –r)

echo `uname –r` (to wypisze wynik czyli numer wersji jądra)

echo ~ (pokazuje katalog domowy aktualnego usera)

PODAWANIE PARAMETRÓW:

-p (forma skrócona)

--parametr (forma pełna)

forma skrócona daje możliwość łączenia kilku paramterów na raz np. netstat -ant
to oznacza –a –n –t a nie słowo ant

#which cat (zwraca informację skąd zostanie odpalony cat tutaj /bin/cat)

URUCHAMIANIE SKRYPTÓW (5 sposobów):

#cat polecenia |bash

#bash polecenia

#sh polecenia

 # ./polecenia (ale najpierw **#chmod u+x** polecenia)

#/pełan_sciezka/polecenia (j.w)

SKRYPT polecenia:

vim polecenia

#!/bin/sh

#komentarze

echo „ten skrypcik cos tam robi ...”

. plik_dolaczany_do_skryptu.sh (include, wstawianie treści zapis: od kropki)

echo „Dzisiaj jest:”`date +%A` (pełna nazwa dnia tygodnia: poniedziałek)

echo „Dzisiaj jest:”`date +%a` (skrótem nazwa dnia tygodnia: pon)

imie =Piotr (zmienna imie)

echo $imie (wyświetla Piotr)

echo ${imie}ek (wyświetla Piotrek)

FORMAT=+%Y-%M-%d

DATA=`date $FORMAT`

tar –czvf archiwum${DATA}.tar.gz katalogi

PRZEKAZYWANIE ZMIENNYCH $1 $2 $3 $4 itd:

vim skrypt.sh

#!/bin/sh

echo $@

shift

echo „mój drugi argument=”$1

#./skrypt.sh ala ma kota (zwroci „ma” ponieważ jest shift)

$@ (zawiera wszystkie argumenty zwróci: ala ma kota)
$? (zawiera kod błędu, 0-oki, 1,2,3... - porażka)
$# (podaj ilość podanych argumentów , tutaj 3)
$0 (zwraca samą swoją nazwę, tutaj: skrypt.sh)
shift (przesuwa parametry w lewo tzn $1 znika $2->$1, $3->$2 itd.)
sleep 2 (pauza 2 sekundowa)
***** (oznacza dowolną ilość dowolnych znaków)
`polecenie` jest równoważne z zapisem $(polecenie)
~ (w to miejsce powłoka wstawia nazwę aktualnego usera)
{nap1,nap2,nap3} (w to miejsce powłoka wstawi kolejne napisy)
{1..10} (w to miejsce powłoka wstawi kolejne liczny)
$((wyrażenie arytmetyczne)) (w to miejsce powłoka wstawi wynik wyrażenia np. $((3*7)) =21
! polecenie (negacja stosowana w warunkach if)
exit wartość (skrypt zwróci wartość np. exit 1)
&& oraz (oba muszą być true)
|| lub (wystarczy że jedno spełnia)

np. ncal 0
echo $? (zwraca 64)

#ls –l `which cat` zapis równoważny z **#ls –l $(which cat)**
#touch nic{1..5}.txt (założy automatycznie 5 plików nic1.txt, nic2.txt itd. Do 5)

INSTRUKCJE WARUNKOWE if:
vim kasowanie

```
#!/bin/sh
if  rm /etc/passwd
        then
                echo ‘udalo sie’
        else
                echo ‘nie udalo sie skasowac’
fi
```

vim sprawdzacz

```
#!/bin/sh
if  which ncal >/dev/null  2>&1
        then
                echo ‘masz ncala’
        else
                echo ‘nie masz ncala’
fi
```

0 – true

1- false

FUNKCJE:

```
Function nazwa() {
        Polecenia
                }
```

#alias dir=’ls –la’

#dir

#listuj() { ls; }

#listuj (użycie funkcji analogiczne do aliasu)

#mdcd () { mkdir $1; cd $1; } (możma wrzucić do .bashrc)

użycie analogiczne do dwóch poleceń: mkdir katalog + cd katalog

#mdcd katalog

przykład funkcji fork() bomb

:( ) { :| :& }; :

PĘTLE:

```
While warunek; do
        Polecenia
Done
```

Until warunek; do

Polecenia

Done

For nazwa in słowa; do

Polecenia

Done

#for i in {1..100}; do echo $i; done (wypisze 100 liczb)

#for i in ala ma asa; do echo $i; done (wypisze: ala , ma , asa)

#for i in {1..100}; do cp 1.txt 1_$i.txt; done (rozmnazanie plikow)

ZMIENNE ŚRODOWISKOWE:

#plik=nic.txt (zmienna lokalna)

#echo $plik (działa tylko lokalnie w sesji)

#export plik=nic.txt (to powoduje globalizację zmiennej środowiskowej)

#env (pokazuje wszystkie zmienne środowiskowe)

#set (j.w. jeszcze więcej danych)

#export PS1='C:/>' (zmienia znak zachęty PS1 standardowo PS1='\h:\w\$')

#echo $PATH (wyświetla ścieżkę poszukiwań)

#echo $PS1 (wyświetla znak zachęty)

/etc/skel/ (domyślny szkielet przy zakładaniu nowego usera, tu wprowadzamy zmiany)

w katalogu domowym każdego usera są ukryte pliki konfigurcyjne .bashrc .profile .bash_logout pobierane z szablonu /etc/skel)

.bashrc – odpala się w oknie terminala, tu jest zmienna PS1

bardzo niebezpieczny jest zapis w .profile

export PATH=.:$PATH (kropka powoduje że mogą uruchomić się z bieżącej lokalizacji więc passwd wcale nie musi być tym z /usr/bin/passwd

wtedy polecenie odpali się od razu bez użycia #./polecenie

/etc/bash.bashrc (tu możemy wstawiać dane globalne dla wszystkich)

np. EDITOR=mcedit (domyślny edytor)

POLECENIA STANDARDOWE

cd (przenosi mnie natychmiast do mojego katalogu domowego)

cd ~/podkatalog1 (przenosi do katologu i podkatalogu domowego bieżącego usera)

cd ~darek (przenosi do katologu domowego usera: darek)

cd - (wraca do poprzedniego katalogu, taki Switch)

cd .. (przenosi piętro wyżej w drzewie katalogowym)

pwd (pokazuje w jakim aktualnie jestem katalogu)

ls ~kurs/fir* (listuje zawartość podkatalogu firma z katalogu domowego usera: kurs)

ls –d ~kurs/fir* (pokazuje nazwę katalogu: /home/kurs/firma zamiast zawartości)

ls –l (więcej informacji tzn:

literka oznaczająca typ pliku, dziewięć liter oznaczających prawa dla właściciela grupy i pozostałych, liczba twardych dowiązań, użytkownik do którego należy plik, grupa do której należy plik, rozmiar, data ostatniej modyfikacji, nazwa)

ls –la (pokazuje również pliki ukryte zaczynające się od kropki)

ls –lS (sortuje listę po rozmiarze od największego)

mkdir karpaty/obrazki (zakładamy od razu podkatalog w katalogu karpaty)

rm –r katalog (kasuje katalog wraz z podstrukturą, nawet pusty

rm –r / (UWAGA to polecenie zniszczy wszystko)

cp ../katalog/* . (kopiuje wszystkie pliki do bieżącego katalogu <kropka>)

cp plik ~kurs (kopiuje plik do katalogu domowego usera kurs)

ls zmienne/ = **ls ./zmienne** (zapis rownoznaczny)

mv *kar*.jpg karpaty/obrazki/ (przesuwa pliki obrazków z karpat)

cat /dev/mouse1 (ruszając myszą widać zapis do tego katalogu)

cat /dev/zero (generuje same zera)

cat /dev/null (czarna dziura)

touch plik.txt (tworzy pusty plik.txt, jeżeli istniał to zmienia jego datę modyfikacji na aktualną)

alternatywnie można użyć **#> plik.txt**

#cat > plik.txt

napisy

<CTRL>D (zakończenie edycji)

ls |more (stronnicowanie informacji, wyjście to q)

ls |less (podobnie jak more ale można przewijać kursorami góra i dół)

ls |sort (posortuje pliki)

cat liczby.txt |sort –n (posortuje numerycznie)

ln –s kalendarz.txt skrotsymboliczny.txt (tworzy link symboliczny o nazwie skrotsymboliczny.txt ze wskazaniem na plik kalendzrz.txt)

ln kalendarz.txt skrottwardy (tworzy twarde dowiązanie

ldd /bin/cat (pokazuje z jakich bibliotek korzysta dany program, odpala na chwilę cat)

kill –l (listuje spis dostępnych sygnałów wysyłanych do procesów, procesy się komunikują między sobą)

ps –ef |grep ssh (wyświetla listę procesów tu zwiazanych z ssh)

nc (netcat przydatne narządzie diagnostyczne!)

#nc –l –p nr_portu (słucha na porcie nr_portu)

TTY1

#nc –l –p 5000

TTY2

#netstat –ant (widać że słucha na porcie 5000)

#nc –v 127.0.0.1 5000

(widać że to co napiszemy w TTY1 pokazuje się na TTY2)

#nc –v 127.0.0.1 80 (sprawdzenie czy działa serwer www)

NADAWCA:

#tar czvf - /usr/share/doc/plik.txt | nc 10.0.2.131 9001

ODBIORCA:

#nc –l –p 9001 |tar –xzvf -

mkfifo scieżkadopliku (tworzy rurkę , kolejkę która czeka aż ktoś ją odczyta)

musimy być w tym samym katalogu (ls –l pokazuje p że jest to pipe)

TTY1

#mkfifo rureczka

#cat rureczka

TTY2

#ls >rureczka

#df –h (pokazuje dostępne miejsce na dysku)

#df –k (j.w w KB)

GRUPOWANIE POLECEN:

#(echo ‘ala’ ; echo ‘ma asa’) |less

PRAWA I UŻYTKOWNICY:

#groupadd grupa (dodajemy nową grupę)

#groupdel grupa (usuwa grupę)

#gpasswd –a user grupa (dodaje usera do grupy)

#gpasswd –d user grupa (usuwa usera z grupy)

less /etc/group (wykaz grup)

less /etc/gshadow (hasła dla grup)

#usermod –G grupa użytkownik (dodawanie użytkownika do grupy - zastąpienie)

#usermod –aG grupa użytkownik (dodawanie użytkownika do grupy - dodanie)

#usermod –s /bin/false user (ustawienie nowej powłoki dla usera – zablokowanie dostępu)

spr: less /etc/passwd

#pwck (sprawdzanie spójności passwd+shadow)

#id user (pokazuje dane usera lub grupy, uid gid itp,)

#groups user (pokazuje do jakich grup przynależy user)

#whoami (wypisuje nazwę aktualnego usera)

#who (wypisuje listę zalogowanych userów)

#w (j.w)

#finger kurs (wypisuje dane o wskazanym userze, domyślnie brak więc: apt-get install finger)

#finger (wypisuje ciekawe informacje o zalogowanych userach)

#useradd –m użytkownik (dodanie nowego użytkownika i jego katalogu domowego o tej samej nazwie)

-m (wymusza założenie katalogu domowego)

-c „pełna nazwa” (ustawia pełną nazwę)

-d <katalog domowy> (ustawia ścieżkę do katalogu domowego, standard: /home)

-g <grupa początkowa> (ustawia przynależność do grupy)

-G <lista_grup> (dodaje usera do wymienionych grup)

-k <szkielet> (wskazuje źródło skąd wkopiować pliki profilu startowe do katalogu domowego, domyślnie /etc/skel)

-s <shell> (ustawia powłokę, domyślnie /bin/sh)

(domyślne wartości są pobierane z /etc/default/useradd)

#userdel użytkownik (usunięcie użytkownika)

#userdel –r użytkownik (usunięcie użytkownika i jego katalogu domowego)

#passwd użytkownik (zmiana hasła użytkownika, root może każdemu)

#chage –d 0 kurs (hasło usera kurs musi być zmienione przy najbliższym zalogowaniu)

#chage –l kurs (listuje ustawienia hasła dla usera kurs)

#chmod u+rw plik.txt (nadaje prawo rw właścicielowi do plik.txt)

#chmod o-x program.exe (odbiera prawo wykonywalności programu pozostałym użytkownikom)

#chown user /sciezka/plik.txt (zmienia właściela plik.txt na user)

#chown user.grupa plik (można od razu zmienić właściciela i grupę)

#chown .grupa plik (zmienia tylko grupę dla plik)

przy kasowaniu pliku decydują uprawnienia do KATALOGU w którym jest ten plik

WYRAŻENIA REGULARNE:

#cat pliktekstowy |grep „ala” –color=AUTO (ładnie podświetla to co pasuje do filtra)

#cat pliktekstowy |grep „a.a” (znajduje a<dowolny_znak>a) np. ustalamy, sama, ma asa itp.

Jeżeli chcemy pozbawić kropki (.) jej specjalnego znaczenia to należy użyć \.

#cat pliktekstowy |grep „a\.” (znajdzie np. www.republika.pl)

#cat pliktekstowy |grep „a[abcdefghijklmnoprstwuz]a” (znajduje w środku literę) np. czasami

#cat pliktekstowy |grep „a[a-zA-Z]a” (jak wyżej ale bez względu na duża czy mała) np.działanie

#cat pliktekstowy |grep „a[0-9]a” (w środku dowolna jedna cyfra) np. a9a ale już nie a12a

#cat pliktekstowy |grep „a[11-15]a” (taki zapis to 1 lub 1-1 lub 5)

#cat pliktekstowy |grep „a[[:alpha:]]a” (w środku dowolny znak alafanumeryczny) np. a9a ala

#cat pliktekstowy |grep „a[0-9]*a” (wystąpi dowolną ilość razy w tym zero razy) np. a123a a9a zaawansaowany

#cat pliktekstowy |grep „a[0-9]\+a” (dowolną niezerową ilość razy) np. a123a a0a a9a

#cat pliktekstowy |grep „pyth\?on” (h\? może wystąpić ale nie musi) python pyton

#cat pliktekstowy |grep „a\{3,5\}” (literka a wystąpi od 3 do 5 razy)

#cat pliktekstowy |grep „pyton\|python” (\| oznacza lub czyli pyton lub python)

#cat pliktekstowy |grep „a\|b[0-9]\+” (a lub b a potem dowolna choć jedna cyfra)

#cat pliktekstowy |grep „\(a\|b\)[0-9]\+” ((a lub b) a potem dowolna choć jedna cyfra)

#cat pliktekstowy |grep „^ala” (znajduje wiersz zaczynający się od ciągu ala) np. ala.....

#cat pliktekstowy |grep „ala$” (znajduje wiersz kończoncy się na ciągu ala) np. .....ala

#cat pliktekstowy |grep „\<ala” (znajduje początek słów zaczynających się od ala)

#cat pliktekstowy |grep „ala\>” (znajduje koniec słów kończoncych się od ala) np. pozwala

\. Pozbawia kropki jej specjalnego znaczenia a\.a oznacza ciąg „a.a”

[] zakres znaków

. dowolny znak

* dowolną ilość razy

\+ dowolną niezerową liczbę razy

\| lub

znak\? ten znak może wystąpić (ale tylko raz) albo nie

a\{3\} oznacza literę a powtórzoną 3 razy

\< początek słowa

\> koniec słowa

grep –n (wypisuje na początku numer linii)

#grep [0-9] (bierze z wej i wypisuje na wyj tzn ze to napiszemy z klawiatury zostanie odfiltrowane)

#cat pliktekstowy | sed „s/tpnet/tpsa/g” |vim - (zastąp w tekście tpnet na tpsa g-zmieni wszystkie w wierszu jeśli wystąpiły kilka razy)

MODUŁY:

#lsmod (pokazuje jakie moduły są załadowane)

#rmmod lp (odładowanie modułu „lp”)

#modinfo /lib/modules/kernel/drivers/char/lp.ko (pełne informacje o module i jego zależnościach)

#insmod /lib/modules/kernel/drivers/char/lp.ko (załadowanie modułu lp.ko najpierw trzeba załadować moduły od których ten zależy)

#dmesg |less (po załadowaniu warto zobaczyć komunikaty jądra i listę modułów załadowanych)

#lsmod |grep lp

#modprobe lp (zamiast insmod , jest preferowane bo ładuje wszystko co wymaga)

konfiguracja jest w katalogu /etc/modprobe.d/

#depmod –a (zbiera informacje o zależnościach miedzy pakietami)

#lspci (sprawdza co jest podłaczone do magistrali PCI)

#lspci -n (odpytuje urządzenia na PCI)

na tej podstawie można prognozować kod VENDOR:DEVICE (10EC:8139)

ls /usr/share/misc(hwdata)/pci.ids

#lsusb (sprawdza co jest podłaczone do USB)

apt-get install strace (śledzi wywołania systemowe i sygnały)

 # strace cat /home/tralalla (zwróci –1 open gdy brak pliku)

 # strace –eopen cat /home/tralalla (j.w. ale bardziej precyzyjne)

apt-get install smartmontools (monitorowanie urządzeń S.M.A.R.T)

#smartctl –a /dev/hda (pokazuje wszelkie info o dysku)

jeśli zobaczymy SMART Disabled to:

#smartctl –s on /dev/hda (włącza technologię SMART)

(jeśli wspiera to można odczytać parametry kondycyjne dysku twardego np. temp. Lub Reollocated_Sector_Count -przeniesione bloki w obaszr awaryjny, powinno być VALUE:100 RAW_VALUE:0)

<u>ARCHIWA TAR:</u>

#tar c firma >firma.tar (składowanie do jednego pliku)

#tar x firma <firma.tar (wykładanie z jednego pliku)

#tar c firma |gzip >firma.tar.gz (j.w. + kompresja)

GZIP:

#tar –czvf firma.tar.gz firma (archiwizuje katalog FIRMA)

-c (create twórz)

-z (gzip kompresuj)

-v (verbose informuj)

-f (file zapisz do pliku)

#tar –xvf firma.tar.gz (wypakuj z archiwum, -x eXtract)

BZIP2:

#tar –cjf firma.tar.bz2 firma

#tar –xjf firma.tar.bz2

w praktyce:

#tar –czvf ~/archiwum`date +%Y-%M-%d`.tar.gz katalogi/

(powstanie w katalogu domowym archiwum2009-05-04.tar.gz)

#file firma.tar.gz (pokazuje co to za plik)

#apt-get install tree

#tree (pokazuje drzewo katalogów w bieżącym miejscu w pseudograficznej formie)

#md5sum * (wylicza sumy md5 dla plików z bieżącego katalogu)

#md5sum * |cut –d" „ –f1 |sort |uniq (wycina 1 kolumnę unikatowych sum)

SUDO:

#apt-get install sudo

/etc/sudores (plik z uprawnieniami dla polecenia sudo, do edycji używamy: **visudo**)

przykład zapisu w /etc/sudores:

kurs ALL=(root) /bin/cat

<kto> <na jakich komputerach>=<jako kto> <jakie może wykonać polecenie (parametry)>

#cat /etc/shadow (użytkownik kurs nie może)

#sudo cat /etc/shadow (z uwagi na wpis sudores już może wykonać to polecenie)

zapis w /etc/sudores

kurs ALL=(ALL) ALL (pozwala zrobić userowi kurs wszystko bez dawania mu hasła roota)

kurs ALL=(ALL) NOPASSWD:ALL (dodatkowo nie pyta o hasło)

przykład zastosowania:

kurs ALL=(ALL) /usr/bin/passwd

#sudo passwd guest

#sudo su - (przełącza nas na konto roota zachowując zmienne środowiskowe)

#sudo su; whoami (whoami wykona się dopiero po zakończeniu sesji su)

%grupa ALL=(root) /usr/bin/vim /etc/motd

(nadajemy grupie uprawnienia do modyfikacji pliku /etc/motd z poziomu vim'a. Taki zapis jest niebezpieczny ponieważ z poziomu vim'a można zrobić :sh i buszować w shellu z prawami roota dlatego powinniśmy stosować inny edytor: **sudoedit** jakponiżej)

%grupa ALL=(root) sudoedit /etc/motd

Nadanie prawa gościowi do zrobienia cat na dowolnym pliku z etc poza /etc/motd:

guest ALL=(ALL) /bin/cat /etc/.* , !/etc/motd

Nadanie userowi guest prawo do restartu serwera ssh:

guest ALL=(root) /etc/init.d/sshd restart (visudo)

potem można to wykonać:

#sudo /etc/init.d/sshd restart

SYSTEM PLIKÓW:

#ldd /bin/cat (pokazuje jakich bibliotek wymaga dany program)

#strace cat plik.txt (pokazuje co dzieje się na styku program-jądro)

np. #strace cat plik 2>out.strace

#vim out.strace (mamy info w kolorach, czytelniejsze)

#stat plik.txt (dokładne info więcej niż ls –la oraz daty modyfikacji)

#file plik.txt (zwraca rodzaj pliku)

#cat /proc/filesystems (pokazuje aktualnie czynne systemy plików)

#ls /lib/modules/`uname –r`/kernel/fs (pokazuje wszystkie systemy plików które obsłuży linux)

#ls –i (pokazuje numery węzłów tzw i-node)

wyjaśnienie

ln –s plik links (dowiązanie symboliczne)

ln plik linkt (dowiązanie twarde)

ls – i

200 plik 200 linkt 201 links

#mount (pokazuje co mamy zamontowane dysk - partycja)

#dumpe2fs /dev/sda7 |less (statystyka systemu plików, widać jakie mamy block size)

#mkfs.ext3 (służy do tworzenia systemu plików, najlepiej zobaczyć: mkfs.<TAB>)

Przykład użycia:

Tworzymy plik_z_zerami 52MB

#dd if=/dev/zero of=plik_z_zerami bs=100K count=512

#mkfs.ext2 plik_z_zerami (format pliku_z_zerami, tworzymy system plików w pliku)

#tune2fs –m0 plik_z_zerami (likwidujemy 5% zarezerwowanych bloków dla superusera)

 # mount –o loop plik_z_zerami /mnt (podmontowujemy nasz stworzony system plików)

/dev/loop0 na /mnt type ext2 (rw)

#losetup /dev/loop0 (pokazuje co to fizycznie jest)

#df –h (sprawdzenie ile mamy miejsca na tych dyskach „plik_z_zerami" również)

#du –sh plik.txt (pokazuje ile zajmuje dany plik)

#umount /mnt/ (odmontowanie w celu dołożenia jurnala do systemu plików)

#tune2fs –j plik_z_zerami (ext2 → ext3)

#mount –o loop plik_z_zerami /mnt (ponowne zamontowanie)

#mount (sprawdzenie)

jeśli chcemy ustawić na stałe montowanie odpowiednich partycji to ustawiamy to:

#vim /etc/fstab

budowa fstab:

co – gdzie – file system – opcje - info dla dump – info dla fsck

#fsck -py urządzenie (naprawa systemu plików bez pytania)

np. fsck –py /dev/sda1 (powinno być wykonywane po odmontowaniu sda1)

FORMATOWANIE:

#mkfs.xfs –f plik_z_zerami

#apt-get install dosfstools

#mkfs.vfat plik_z_zerami (format FAT)

#mkfs.vfat -F 32 plik_z_zerami (format FAT32)

#mkfs.ext3 plik_z_zerami (format EXT3)

SWAP

#free (pokazuje zajętość RAM)

#cat /etc/fstab (szukamy typu swap)

#swapoff /dev/sda2 (wyłączenie SWAPa)

#mkswap /dev/sda2 (format SWAPa)

#swapon –a (włączenie wszystkich SWAPów)

#swapon –s (pokazuje jakich urządzeń wymiany używamy)

CDR:

Tworzenie obrazu ISO 9660 w standardzie –J joliet –r zachowanie uprawnień, -o output do pliku doc.iso ze wskazanego miejsca)

#apt-get install mkisofs (instalacja paczki)

#mkisofs -J –r –o doc.iso /usr/share/doc (tworzenie obrazu ISO, starsza metoda)

#genisoimage –o tworzonyobraz.iso /home (tworzy z danego katalogu obraz ISO)

#apt-get install wodim (instalacja programu do zapisywania na CDR)

#wodim doc.iso (wypalenie na płycie CDR)

#mount –o loop doc.iso /mnt/ (podmontowanie obrazu ISO do /mnt)

#mount (sprawdzenie stanu zamontowanych partycji)

QUOTA

#apt-get install quota (instalacja paczki Quota)

krok1:

należy zamontować system plików z opcją “usrquota” wpis do **/etc/fstab**

/dev/sda9 /home ext3 defaults,**usrquota** 1 1

#mount –o remount /home (przemontowanie partycji)

#mount (widać skutek w zapisie: rw,usrquota)

krok2:

tworzenie pliku z informacjami o limitach, najlepiej robić w trybie singeluser

#runlevel (N 2)

#init 1 (żeby nikt nam nie mieszał na dysku w tym czasie)

#quotacheck –cm /home (-c stwórz plik, -m nie przemontowuj systemu plików)

w wyniku tej operacji powstaje plik w: /home/***aquota.user***

#init 2 (powrót do standardowego runlevela)

krok3:

właczamy quotę dla danego systemu plików

#quotaon /home

krok4:

ustawianie limitów dla userów:

#edquota –u student (edycja limitów dla usera: student)

blocks – ile kilobajtów zajmuje aktualnie user student w systemie plików /dev/sda9

soft – mięki limit ilości kilobajtów

hard - twardy limit ilości kilobajtów

inodes – ilość iwęzłów zajmowanych aktualnie przez danego usera

soft – mięki limit ilości iwęzłów

hard - twardy limit ilości iwęzłów

#export EDITOR=vim (jeśli chcemy edytować w swoim ulubionym edytroku)

#quota (pokazuje na razie że ograniczeń brak)

ustawiamy teraz limit mięki na 10MB a twardy na 20MB

/dev/sda9 904 **10000 20000** 152 0 0

krok5: sprawdzenie jak to wszystko działa

po zalogowaniu się na user:student polecenie #quota pokazuje wprowadzone limity

* wskazuje na przekroczenie limitu miękkiego, można to przetestować:

$cat /dev/zero |head –c 1000000 >duzyplik1.txt

$quota (statystyki dla studenta)

jako root można przesłać ostrzeżenie na mail: /var/mail/student

#warnquota

$cat /dev/zero >duzyplik2.txt

błąd zapisu: przekroczony limit dyskowy

$quota

$du –hs /home/student (widać że osiągneliśmy górny limit 20MB)

#repquota –a (sprawdzenie limitów dla wszystkich userów na urządzeniu /dev/sda9)

SZTUCZKI KRUCZKI:

Stworzyć plik o rozmiarze 1 Bajta (3 sposoby):

#echo > nic.txt

#echo –n 1 >nic.txt

#echo /dev/zero |head –c 1 >nic.txt

w ostatnich 1000 liniach logów obliczyć ile było odwiedziń IE:

 # cat apache_access.log |tail –n 1000 |grep MSIE |wc -l

jaki dzień tygodnia wypadł na ostatnią modyfikację największego pliku w zadanym katalogu:

#date +%A - -date=`ls –S - -full-time /katalog/ |head –n 2|tail –n 1|cut –d' ' –f6`

(zwróci konkretną nazwę dnai tygodnia)

Stwórz wyrażenie regularne pasujące do wzorca adresu IP:

#grep „\([0-9]\{1,3\}\.\)\{3\}\([0-9]\{1,3\}"

Jak najszybciej zablokować użytkownika bez jego kasowania:

w pliku **/etc/passwd** zmienić shell na **/bin/false** lub /bin/nologin (to samo można dokonać zamiast vim /etc/passwd poleceniem **#usermod –s /bin/false user**)

Oblicz ile jest wszystkich plików na dysku:

#find / -type f |wc –l

Wymuś zmianę hasła przy pierwszym zalogowaniu:

#chage –d 0 user

Jak zawiesić linuxa:

#( ) { :|: & };:

Sztuczki plikowe:

#touch nazwa{1..10}.txt (zakłada 10 plików od nazwa1.txt do nazwa10.txt)

#touch ala ma kota.txt (założy 3 pliki: ala, ma, kota.txt)

#touch ala\ ma\ kota.txt (załozy 1 plik ala ma kota.txt)

**#touch 'ala \ ma 10$usa*.txt'** (założy 1 plik ala \ ma 10$usa*.txt)

**#mkdir ** (po \ spacja spowoduje założenie niewidocznego katalogu) nazwa to spacja

#touch ... (załozy niewidoczny dokument)

#touch a/b.txt (tworzy plik b.txt w katalogu a)

Przekierowanie standardowego wyjścia błędów na standardowe wejście:
2>&1

Jak wygenerować bardzo duży plik:
#dd if=/dev/zero of=duzyplik bs=1M count=1 seek=1023
#ls –lh duzyplik (zwraca 1GB)
#scp duzyplik student@127.0.0.1:
#cpipe (pipemeter)

Jak zobaczyć jakie mamy partycje w systemie
 # cat /proc/partitions (plik pokazuje wszystkie partycje jakimi dysponujemy)
#fdisk /dev/had (na tej podstawie wiemy jaki mamy dysk hda lub sda)
m - help
p - lista partycji
#mount (listuje wszystkie zamontowane partycje)

Tworzenie obrazu płyty iso:
Wkładamy płytę CD
#mount (rozpoznajemy cdrom /dev/sr0)
#cat /dev/sr0 > debian.iso ; eject (tworzymy obraz płyty i po zakończeniu wysuń)
#wodim debian.iso (write optical disk image medium – wypalenie na CDRW)
cdrecord (alternatywny programik do wypalania)

Odzysk hasła root'a:
 # cat /proc/partitions (sprawdzamy jakie dyski są wykryte)
#fdisk /dev/sda ->p (szukamy partycji bootującej ozn: *)
#mkdir /mnt/ratunek
#mount /dev/sda1 /mnt/ratunek (podmontowujemy partycję bootującą)
są dwie metody: PAM lub /etc/shadow
ad.1 uruchamiamy bootującą płytę z Knopixem
useradd test1
passwd test1
cat /etc/shadow (mamy już skrót hasła)
vim /etc/shadow
<v> :w schowek.txt
vim /mnt/ratunek/etc/shadow

:r schowek.txt (w miejsce hasła root: :)

:w! (wymuszamy zapis na podmontowanej partycji)

ad.2

vim /etc/pam.d/login

auth sufficient pom_permit.so (wtedy nie pyta o hasło)

Przykładowy skrypt podstawowego firewall'a:

#!/bin/bash

iptables -F

iptables -t nat -F

iptables -A INPUT -i lo -j ACCEPT

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT

iptables -A INPUT -s 192.168.1.0/24 -p tcp -m tcp --dport 22 -j ACCEPT

iptables -A INPUT -p tcp -s 192.168.1.0/24 -m tcp --dport 80 -j ACCEPT

#iptables -A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT

#iptables -A INPUT -m limit --limit 1/sec -j LOG

iptables -t nat -A POSTROUTING -o eth0 -s 192.168.1.1/24 -j SNAT --to-source 10.0.2.100

iptables -t nat -A PREROUTING -p tcp -m tcp --dport 22 -j DNAT --to 192.168.1.123

iptables -P INPUT DROP

MACIERZE:

RAID (Redundant Array of Independent Disks)

RAID 0 – striping (przyspieszenie)

RAID 1 – mirroring (bezpieczeństwo)

Macierze dyskowe widoczne są jako urządzenia **/dev/mdX** (X-liczba) Każde takie urządzenie odpowiada pojedyńczej partycji na której można założyć system plików. Programy instalacyjne na ogół potrafią zbudować macierz. Do obsługi macierzy w linuxie służy polecenie **mdadm**

#apt-get install mdadm

#mdadm - -create –l <pozim> -n <iloscdyskow> - -auto=yes /dev/mdX <urzadzenia>
(tworzenie nowej macierzy, wśród urządzeń może wystąpić słowo „missing")

#mdadm - -assemble <urzadzenie md> <urzadzenia skaldowe> (odbudowa już istniejącej macierzy ze wskazanych urządzeń)

#mdadm - -add <urzadzenie md> <dysk> (dodanie dysku do istniejącej macierzy)

#mdadm - -fail <urzadzenie md> <dysk> (oznacz dysk jako uszkodzony)

#mdadm - -remove <urzadzenie md> <dysk> (usuwa dysk z macierzy)

#mdadm - -monitor <urzadzenie md> (monitoring zdarzeń)

Aktualny stan macierzy w systemie operacyjnym raportowany jest w pliku **/proc/mdstat**

Informacje o systemie plików i punktach montowania:

#cat /proc/mdstat

#cat /proc/partitions

#cat /etc/mtab

#cat /etc/fstab

#mount

#df –h

Stworzenie RAID1 na dwóch partycjach /dev/sdc1 i /dev/sdd1 jako /dev/md1

#mdadm –C –v /dev/md1 - -level=1 –n 2 /dev/sdc1 /dev/sdd1

PRZYKŁAD PRAKTYCZNY:

Założenie że mamy dwa dyski sda i sdb. Na dysku /dev/sda jest już zainstalowany system, dołaczamy drugi czysty dysk /dev/sdb. Mamy już zainstalowaną paczkę mdadm:

#apt-get install mdadm

W testowym przykładzie mamy następujący podział na partycje:

/dev/sda1 /boot (83)

/dev/sda2 / (83)

/dev/sda3 swap (82)

/dev/sda4 /home (83)

#fdisk –l /dev/sda (dokładny listing partycji dysku sda)

Na drugim dysku tworzymy identyczne rozmiary partycji ale typu fd – Linux raid autodetect, wszystko poleceniem: fdisk (m-help)

/dev/sdb1 /boot (fd)

/dev/sdb2 / (fd)

/dev/sdb3 swap (fd)

/dev/sdb4 /home (fd)

#fdisk –l /dev/sdb (dokładny listing partycji dysku sdb)

Powołujemy macierz do życia dla każdej partycji MATRIX RAID1

#mdadm - -create /dev/md0 - -level 1 - -raid-devices=2 missing /dev/sdb1

#mdadm - -create /dev/md1 - -level 1 - -raid-devices=2 missing /dev/sdb2

#mdadm - -create /dev/md2 - -level 1 - -raid-devices=2 missing /dev/sdb3

#mdadm - -create /dev/md3 - -level 1 - -raid-devices=2 missing /dev/sdb4

zamiast dysku pierwszego /dev/sda{1..4} użyliśmy słowa missing czyli brak ponieważ na nim istnieje nasz system operacyjny którego nie chcemy uszkodzić. Teraz możemy zobaczyć jak wygląda nasz matrix:

#cat /proc/mdstat

Personalities – rodzaj raidu, pierwszy md0 active na partycji na razie sdb1

[_U] _ brak dysku, U up and running

Teraz formatujemy (ext3 i swap) partycje matrixowe i przenosimy na nie system plików:

#mkfs.ext3 /dev/md0

#mkfs.ext3 /dev/md1

#mkswap /dev/md2

#mkfs.ext3 /dev/md3

#mkdir /mnt_root ; mkdir /mnt_boot ; mkdir /mnt_home

#mount /dev/md0 /mnt_boot

#mount /dev/md1 /mnt_root

#mount /dev/md3 /mnt_home

CZYNNOŚCI KONFIGURACYJNE:

#cat /boot/grub/menu.lst (szukamy zapisu: initrd /boot/initrd.img-2.6.8)

w starszych wersjach:

#vim /etc/mkinitrd/mkinitrd.conf (w starszych wersjach zmieniamy tu)

MODULES=most -> MODULES=dep

ROOT=probe -> ROOT=”/dev/md1 ext3”

#mkinitrd –o /boot/initrd.img-2.6.8-md

w nowszych wersjach mkinitrd zastąpiono mkinitramfs:

#depmod

#lsmod >> /etc/initramfs-tools/initramfs.conf (to spowoduje dopisanie listy modułów do pliku konfiguracyjnego z którego korzysta mkinitramfs)

#vim /etc/initramfs-tools/initramfs.conf (edycja pliku, zostawiamy tylko pierwszą kolumnę z nazwami modułów, moduły scsi lub sata umieszczamy na szczycie listy, usuwamy wpis METADATA=xxxxx z pliku /etc/mdadm/mdadm.conf)

#mkinitramfs –o /boot/initrd.img-2.6.8-md

Edycja GRUBa:

#vim /boot/grub/menu.lst

dopisujemy nową sekcję:

title MATRIX Disk1

root (hd0,0)

kernel /boot/vmlinuz-2.6.8 root=**/dev/md1** ro

initrd **/boot/initrd.img-2.6.8-md**

savedefault

boot

oraz w razie awarii 1 dysku żeby mieć możliwość zastartowania z drugiego:

title MATRIX Disk2

root (hd1,0)

kernel /boot/vmlinuz-2.6.8 root=**/dev/md1** ro

initrd **/boot/initrd.img-2.6.8-md**

savedefault

boot

teraz czas na plik konfiguracyjny macierzy:

#vim /etc/mdadm/mdadm.conf

DEVICE /dev/sda* /dev/sdb*

#mdadm - - detail - -scan >> /etc/mdadm/mdadm.conf (dopisanie z automatu)

Przekopiowujemy dane z dysku pierwotnego na naszą podmontowaną macierz która składa się narazie z jednego dysku /dev/sdb. Do kopiowania najlepiej użyć rsync.

#apt-get install rsync

#rsync –auHx - -exclude=/proc/* - -exclude=/sys/* - -exclude=/boot/* - -exclude=/home/* - -exclude=/mnt_boot/ - -exclude=/mnt_root/ /* /mnt_root

#mkdir /mnt_root/proc /mnt_root/boot /mnr_root/sys

#chmod 555 /mnt_root/proc

#rsync –auHx /boot/ /mnt_boot/

#rsync –auHx /home/ /mnt_home/

Ustawiamy właściwe wpisy w fstab:

#vim /mnt_root/etc/fstab

/dev/sda1 -> /dev/md0

/dev/sda2 -> /dev/md1

/dev/sda3 -> /dev/md2

/dev/sda4 -> /dev/md3

#reboot

Start z MATRIX Dysk2

Po restarcie system startuje na “uszkodzonym” raid1 (bez dysku pierwszego)

Sprawdzamy czy na pewno uruchomił matrix:

#df (widać że mamy /dev/md*)

teraz za pomocą fdisk na dysku pierwszym (pierwotnym) zmieniamy typy partycji na fd (linux raid autodetect) – to niszczy wszystkie dane na tym dysku!

#fdisk –l /dev/sda

/dev/sda1 /boot (fd)

/dev/sda2 / (fd)

/dev/sda3 swap (fd)

/dev/sda4 /home (fd)

Mamy dysk przygotowany do właczenia go do macierzy:

#mdadm /dev/md0 –a /dev/sda1

#mdadm /dev/md1 –a /dev/sda2

#mdadm /dev/md2 –a /dev/sda3

#mdadm /dev/md3 –a /dev/sda4

#cat /proc/mdstat (docelowo powinno być [UU])

na nowo konfigurujemy /etc/mdadm/mdadm.conf (kasujemy w nim wszystko oprócz

DEVICE /dev/sda* /dev/sdb*

#mdadm - - detail - -scan >> /etc/mdadm/mdadm.conf

teraz pozostało zainstalować GRUBa na obu dyskach w MBR by w razie awarii każdy z nich mógł wystartować, w pliku /boot/grub/device.map należy jeszcze dodatkowo umieścić alias

(hd1) /dev/sdb

#grub - -device-map=/boot/grub/device.map

>root (hd0)

>setup (hd0,0)

>root (hd1)

>setup (hd1,0)

>quit

#dpkg-reconfigure mdadm (można ustawić żeby nam wysyłał maila w razie awarii)

SPRAWDZENIE:

- po odłączeniu jednego z dysków system wstaje z informacją dmesg , bootowanie możliwe jest tylko z 1 tego fizycznie podłączonego dysku, df – macierz działa ale tylko na 1

urządzeniu.

- Można usunąć dysk z partycji arrays: **mdadm /dev/md0 –f /dev/sdb1 –r /dev/sdb1**
- Po ponownym podłączeniu dysku w /proc/mdstat nadal mamy status [_U], należy ponownie dołączyć dysk do macierzy: **mdadm /dev/md0 –a /dev/sdb1** (i tak dla wszystkich pozostałych partycji. Statystyki zobaczymy w **cat /proc/mdstat**
- Metoda na grab partition map: **sfdisk –d > sdb-parts.dump**
- Na nowym dysku wtedy robimy**: sfdisk /dev/sdb < sdb-parts.dump**
- Wyłaczenie swap'a: **swapoff /dev/sdb2**
- Włączenie swap'a: **swapon /dev/sdb2**
- Restart maszyny: **shutdown –r now** , docelowo powinno być [UU]

#mdadm /dev/md0 –f /dev/sdb1 (oznaczenie dysku jako fault – status F)

#mdadm /dev/md0 –r /dev/sdb1 (usunięcie dyski z macierzy)

#mdadm /dev/md0 –a /dev/sdb1 (dodanie dysku do macierzy)

CIEKAWE LINKI:

www.sebool.net