# FreeBSD – czyli nie taki diabeł straszny cz. 2

## TLUG - 12.01.2008


FreeBSD®

# O wydajności słów kilka

## Partycjonowanie

Bezpieczeństwo, stabilność:

- Oddzielenie od siebie funkcjonalnych części systemu
- Łatwe backupowanie (np. RAID 1,5 etc.) bądź szyfrowanie jedynie ważnych miejsc.
- W razie uszkodzenia filesystemu nie trzeba odzyskiwać reszty partycji, jedynie uszkodzon
- W przypadku nie korzystania z zapisu na partycji można taka montować w trybie read-only (/boot, /)

Wydajność:

- Częściej wykorzystywane katalogi (read+write >read) mogą być umieszczone bliżej zewnętrznej granicy dysku.
- Dopasowanie rozmiaru bloku do rozmiaru plików docelowych.
- Wykorzystywanie specyficznych cech wydajnościowych filesystemow (np. ext2 do małych partycji)
- Grupując pliki na partycjach wg. charakterystyki dostępu pozwalamy filesystemowi dopasować ustawienia

# Kompilacja kernela – o tym, że można inaczej

```
Plik  Edycja  Widok  Terminal  Zakładki  Pomoc
[root@0xdead /usr/src/sys/i386/conf]# ls
.cvsignore      GENERIC         Makefile        PAE             VIOLETTE
DEFAULTS        GENERIC.hints   NOTES           SMP             XBOX
[root@0xdead /usr/src/sys/i386/conf]# cp GENERIC SAMPLE
[root@0xdead /usr/src/sys/i386/conf]# echo "Konfiguracja ulubionym edytorem tekstu (nastepny slajd)"
Konfiguracja ulubionym edytorem tekstu (nastepny slajd)
[root@0xdead /usr/src/sys/i386/conf]# cd /usr/src
[root@0xdead /usr/src]# make buildkernel KERNCONF=SAMPLE && make installkernel KERNCONF=SAMPLE
```

FreeBSD®

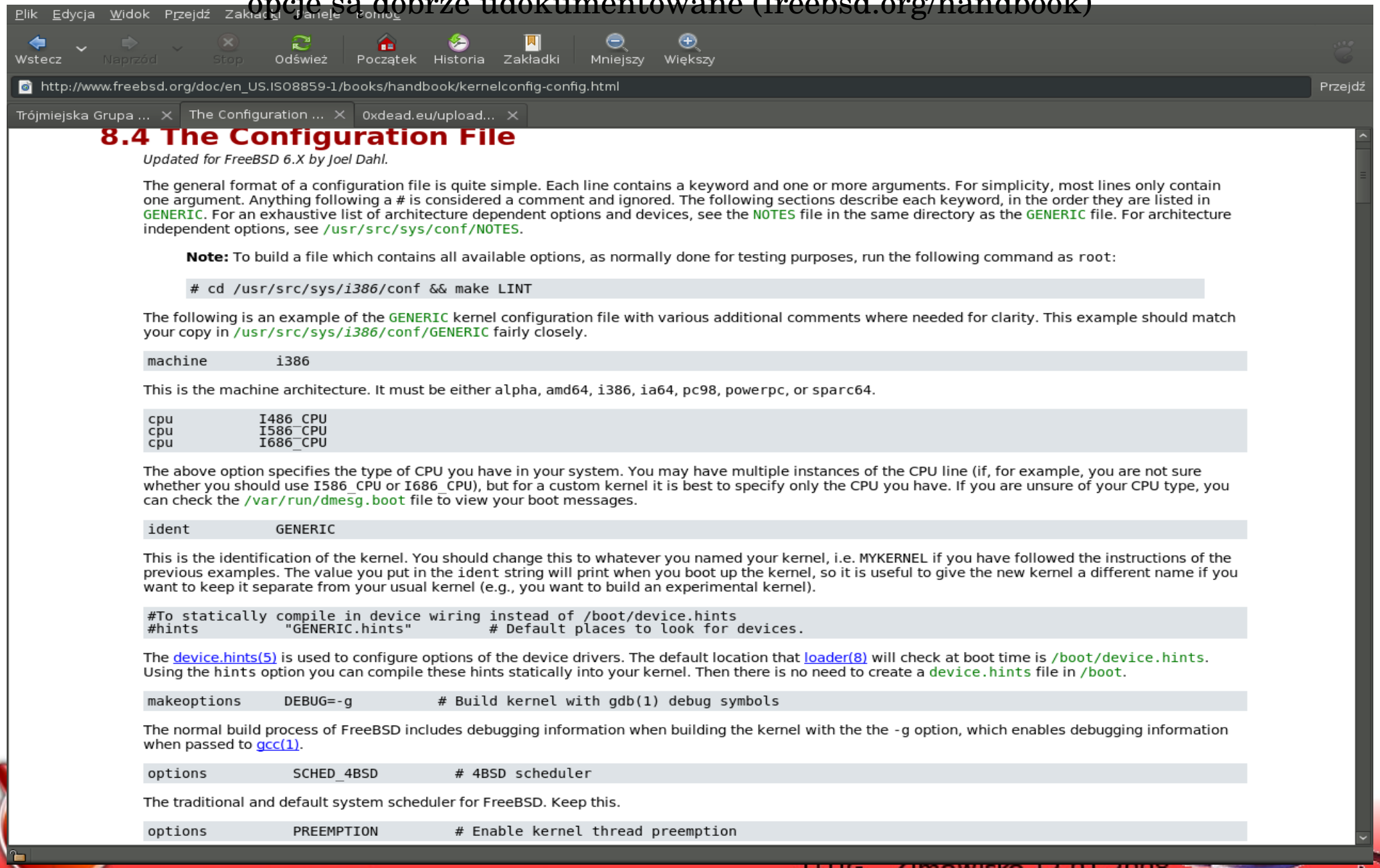
# Plik konfiguracyjny jądra

```
cpu             I486_CPU
cpu             I586_CPU
cpu             I686_CPU
ident           GENERIC

# To statically compile in device wiring instead of /boot/device.hints
#hints          "GENERIC.hints"         # Default places to look for devices.

makeoptions     DEBUG=-g                # Build kernel with gdb(1) debug symbols

options         SCHED_4BSD              # 4BSD scheduler
options         PREEMPTION              # Enable kernel thread preemption
options         INET                    # InterNETworking
options         INET6                   # IPv6 communications protocols
options         SCTP                    # Stream Control Transmission Protocol
options         FFS                     # Berkeley Fast Filesystem
options         SOFTUPDATES             # Enable FFS soft updates support
options         UFS_ACL                 # Support for access control lists
options         UFS_DIRHASH             # Improve performance on big directories
options         UFS_GJOURNAL            # Enable gjournal-based UFS journaling
options         MD_ROOT                 # MD is a potential root device
options         NFSCLIENT               # Network Filesystem Client
options         NFSSERVER               # Network Filesystem Server
options         NFS_ROOT                # NFS usable as /, requires NFSCLIENT
```

FreeBSD®

**Wbrew pozorom konfiguracja jest prosta**

opcje są dobrze udokumentowane (freebsd.org/handbook)




## 8.4 The Configuration File

*Updated for FreeBSD 6.X by Joel Dahl.*

The general format of a configuration file is quite simple. Each line contains a keyword and one or more arguments. For simplicity, most lines only contain one argument. Anything following a # is considered a comment and ignored. The following sections describe each keyword, in the order they are listed in GENERIC. For an exhaustive list of architecture dependent options and devices, see the NOTES file in the same directory as the GENERIC file. For architecture independent options, see /usr/src/sys/conf/NOTES.

**Note:** To build a file which contains all available options, as normally done for testing purposes, run the following command as root:

```
# cd /usr/src/sys/i386/conf && make LINT
```

The following is an example of the GENERIC kernel configuration file with various additional comments where needed for clarity. This example should match your copy in /usr/src/sys/i386/conf/GENERIC fairly closely.

```
machine         i386
```

This is the machine architecture. It must be either alpha, amd64, i386, ia64, pc98, powerpc, or sparc64.

```
cpu             I486_CPU
cpu             I586_CPU
cpu             I686_CPU
```

The above option specifies the type of CPU you have in your system. You may have multiple instances of the CPU line (if, for example, you are not sure whether you should use I586_CPU or I686_CPU), but for a custom kernel it is best to specify only the CPU you have. If you are unsure of your CPU type, you can check the /var/run/dmesg.boot file to view your boot messages.

```
ident           GENERIC
```

This is the identification of the kernel. You should change this to whatever you named your kernel, i.e. MYKERNEL if you have followed the instructions of the previous examples. The value you put in the ident string will print when you boot up the kernel, so it is useful to give the new kernel a different name if you want to keep it separate from your usual kernel (e.g., you want to build an experimental kernel).

```
#To statically compile in device wiring instead of /boot/device.hints
#hints          "GENERIC.hints"         # Default places to look for devices.
```

The device.hints(5) is used to configure options of the device drivers. The default location that loader(8) will check at boot time is /boot/device.hints. Using the hints option you can compile these hints statically into your kernel. Then there is no need to create a device.hints file in /boot.

```
makeoptions     DEBUG=-g            # Build kernel with gdb(1) debug symbols
```

The normal build process of FreeBSD includes debugging information when building the kernel with the the -g option, which enables debugging information when passed to gcc(1).

```
options         SCHED_4BSD          # 4BSD scheduler
```

The traditional and default system scheduler for FreeBSD. Keep this.

```
options         PREEMPTION          # Enable kernel thread preemption
```

# Mechanizm sysctl – użyteczne narzędzie do tuningu

Linux sysctl:

```
[root@magdalene~]# sysctl -a | grep -c ''
794
```

FreeBSD sysctl:

```
[shd@0xdead ~]$ sysctl -a | grep -c ''
1154
```

Dzięki sysctl możemy odczytać/zmodyfikować.

Informacje:

```
[shd@0xdead ~]$ sysctl kern.version
kern.version: FreeBSD 6.2-STABLE #8: Tue Jul  3 09:11:58 CEST 2007
    shd@0xdead.eu:/usr/obj/usr/src/sys/VIOLETTE
```

Parametry:

```
[shd@0xdead ~]$ sysctl kern.maxprocperuid
kern.maxprocperuid: 1357
[shd@0xdead ~]$ sysctl net.inet.tcp.blackhole
net.inet.tcp.blackhole: 2
[shd@0xdead ~]$ sysctl net.inet.ip.ttl=255
net.inet.ip.ttl: 255
[shd@0xdead ~]$ sysctl security.bsd.see_other_uids=0
security.bsd.see_other_uids: 0
[shd@0xdead ~]$ sysctl kern.securelevel
kern.securelevel: -1
```




# System portów – Charakterystyczna cecha fbsd

System portów jest to pakiet skryptów umożliwiający bezproblemową instalację ponad 17000 programów.

| Instalacja z repozytorium | Kompilacja źródeł |
|---|---|
| •Łatwość instalacji<br>•Szybkość – za pomocą jednej komendy<br>•Bezpieczeństwo – wystarczy zaufać repozytorium | •Elastyczność – możliwość dołączenia tylko takich modułów jakie bedziemy potrzebowali<br>•Optymalizacja – dostosowanie kodu maszynowego do używanej przez nas konfiguracji sprzętowej |

## System portów

- Łatwość instalacji – cd /usr/ports/security/sudo && make install clean -> gotowe!
- Szybkość – za pomocą powyższej komendy
- Bezpieczeństwo – Port każdego programu ma listę mirrorów, z których można ściągnąć źródła. Po pobraniu, automatycznie sprawdzane są sumy kontrolne danych. Wszystkie modyfikacje systemu portów są sprawdzane przez społeczność.
- Elastyczność – Najważniejsze opcje kompilacji możemy przekazać w parametrze, bądź za pomocą menu tekstowego (make configure – w przypadku braku konfiguracji uruchamia się samoczynnie)
- Optymalizacja – Binarki budowane są z uwzględnieniem stosownych optymalizacji.

# Rany, gdzie tu iptables?

W skład base systemu wchodzą 3 firewalle:
- ipfw2 (IP FireWall)
- ipf (IP Filter)
- pf (Packet Filter)

ipfw2 – od samego początku pisany przez developerów FreeBSD i dla niego. Posiada zarówno regułki stanowe, jak i bezstanowe. Kontrola przepływu realizowana za pomocą dummynet. Pozwala emulować różne stadia obciążenia sieci, zagubienia pakietów.

ipf – nie pisany pod konkretny system. Można go znaleźć również w NetBSD, OpenBSD, SunOS™, HP/UX, oraz Solaris™. Pierwotnie pisany jako bezstanowy, obecnie zawiera również reguły stanowe. Bardziej zaawansowane funkcje są defaultowo są dezaktywowane co pociąga za sobą wysokie bezpieczeństwo.

Pf – kompletny firewall zawierający kontrolę przepływu i translację adresów zmigrowany z systemu OpenBSD. Posiada moduł normalizacji pakietów.

Jeśli nie przemawia do Ciebie żadna z powyższych opcji w drzewie portów można znaleźć również pakiet iptables


# FreeBSD - czyli nie taki diabeł straszny cz. 2

## Część mojej konfiguracji – przykład IPFW2

Plik Edycja Widok Terminal Zakładki Pomoc

```
for ((y=1; y<=2; y++));
do
        for ((i=1; i<=100; i++));
        do $cmd queue $y$i config weight $i pipe $y;
        done;
done;
echo test
}


#                       FLANK LOCAL MOVEMENT FROM DUMMYNET
$add 90 allow ip from $siec to $siec

#                       DUMMYNET
$cmd pipe 1 config bw 450Kbit/s delay 30
$cmd pipe 2 config bw 50Kbit/s delay 30
                                                                                # server
$add 100 queue ${in}1 tag $queuetag ip from any to me uid mlnet { not tagged $queuetag }                # mldonkey
$add 110 queue ${out}1 tag $queuetag ip from me to any uid mlnet { not tagged $queuetag }
$add 120 queue ${in}5 tag $queuetag ip from any to me { dst-port 80 or dst-port 443 } { not tagged $queuetag }  # http / https
$add 120 queue ${in}3 tag $queuetag ip from any to me uid www { not tagged $queuetag }
$add 130 queue ${out}3 tag $queuetag ip me to any uid www { not tagged $queuetag }
#$add 130 queue ${out}5 tag $queuetag ip from me to any { src-port 80 or src-port 443 } { not tagged $queuetag }
$add 140 queue ${in}3 tag $queuetag ip from any to me 25 { not tagged $queuetag }                       # smtp
$add 150 queue ${out}3 tag $queuetag ip from me 25 to any { not tagged $queuetag }
                                                                                # network
$add 160 queue ${in}10 tag $queuetag ip from any 80,443 to { 10.254.239.0/24 or 192.168.0.0/24 } { not tagged $queuetag }       # http / https
$add 170 queue ${out}10 tag $queuetag ip from { 10.254.239.0/24 or 192.168.0.0/24 } to any 80,443 { not tagged $queuetag }
$add 180 queue ${in}15 tag $queuetag ip from any 25,110,143,995,993 to { 10.254.239.0/24 or 192.168.0.0/24 } { not tagged $queuetag } # e-mail
$add 190 queue ${out}15 tag $queuetag ip from { 10.254.239.0/24 or 192.168.0.0/24 } to any 25,110,143,993,995 { not tagged $queuetag }

$add 200 queue ${in}50 tag $queuetag ip from any to any in recv $wif { not tagged $queuetag }           # default
$add 210 queue ${out}50 tag $queuetag ip from any to any out xmit $wif { not tagged $queuetag }

queueCfg

#                       FIREWALL
$add 410 divert natd ip from any to any in recv $wif
$add 420 divert natd ip from any to any out xmit $wif
$add 430 check-state
$add 440 skipto 500 ip from any to any in recv $wif

$add 450 allow ip from any to any keep-state            #no restrictions outgoing connections

#       Wchodzace
$add 500 allow ip from any to me 22 limit dst-port 20
```

Plik /etc/opt/ipf/ipf.conf:

# Przykład konfiguracji ipf

```
# IP-FILTER rules file

# ============================================================================
# Input packets
# ============================================================================

block in quick on nei0 from 172.16.0.0/12 to any
block in quick on nei0 from 127.0.0.0/8 to any
block in quick on nei0 from 10.0.0.0/8 to any
block in quick on nei0 from 0.0.0.0/32 to any
block in log level local2.alert quick on nei0 from 192.168.11.34/32 to any

# SSH Server
pass in quick on nei0 proto tcp from any to any port = 22 flags S keep state keep frags

# TELNET Server
# pass in quick on nei0 proto tcp from any to any port = 23 flags S keep state keep frags

# FTP Server (Active)
# pass in quick on nei0 proto tcp from any to any port = 21 flags S keep state keep frags

# FTP Server (Passive) (FTP Server (Active) should be uncommented, too)
# pass in quick on nei0 proto tcp from any to any port > 1023 flags S keep state keep frags

block return-rst in log level local2.alert quick on nei0 proto tcp from any to 192.168.11.34/32
block return-icmp-as-dest(port-unr) in log level local2.alert quick on nei0 proto udp from any to 192.168.11.34/32
block in quick on nei0 all

# ============================================================================
# Output packets
# ============================================================================

pass out quick on nei0 proto tcp from any to any flags S keep state keep frags
pass out quick on nei0 proto udp from any to any keep state keep frags
pass out quick on nei0 proto icmp from any to any keep state keep frags
```

Plik /etc/opt/ipf/ipnat.conf:


## Przykład PF

```
altq on fxp0 cbq bandwidth 1.5Mb queue { std_ext, www_ext, boss_ext }

queue std_ext        bandwidth 500Kb cbq(default borrow)
queue www_ext        bandwidth 500Kb { www_ext_http, www_ext_misc }
  queue www_ext_http bandwidth 50% priority 3 cbq(red borrow)
  queue www_ext_misc bandwidth 50% priority 1 cbq(borrow)
queue boss_ext       bandwidth 500Kb priority 3 cbq(borrow)
queue net_int    bandwidth 1.0Mb { std_int, it_int, boss_int }
  queue std_int  bandwidth 250Kb cbq(default borrow)
  queue it_int   bandwidth 500Kb cbq(borrow)
  queue boss_int bandwidth 250Kb priority 3 cbq(borrow)
queue www_int    bandwidth 99Mb cbq(red borrow)
altq on fxp1 cbq bandwidth 100% queue { internal_dmz, net_dmz }
queue internal_dmz   bandwidth 99Mb cbq(borrow)
queue net_dmz        bandwidth 500Kb { net_dmz_http, net_dmz_misc }
  queue net_dmz_http bandwidth 50% priority 3 cbq(red borrow)
  queue net_dmz_misc bandwidth 50% priority 1 cbq(default borrow)
queue internal_dmz
queue net_dmz        bandwidth 500Kb { net_dmz_http, net_dmz_misc }
  queue net_dmz_http priority 3 cbq(red)
  queue net_dmz_misc priority 1 cbq(default)

main_net  = "192.168.0.0/24"
it_net    = "192.168.1.0/24"
int_nets  = "{ 192.168.0.0/24, 192.168.1.0/24 }"
dmz_net   = "10.0.0.0/24"
boss      = "192.168.0.200"
wwwserv   = "10.0.0.100"

block on { fxp0, fxp1, dc0 } all
pass in on fxp0 proto tcp from any to $wwwserv port { 21, > 49151 } flags S/SA keep state queue www_ext_misc
pass in on fxp0 proto tcp from any to $wwwserv port 80 flags S/SA keep state queue www_ext_http

pass out on fxp0 from $int_nets to any keep state
pass out on fxp0 from $boss to any keep state queue boss_ext

pass in on dc0 from $int_nets to any keep state
pass in on dc0 from $it_net to any queue it_int
pass in on dc0 from $boss to any queue boss_int
pass in on dc0 proto tcp from $int_nets to $wwwserv port { 21, 80, > 49151 } flags S/SA keep state queue www_int

pass out on dc0 from dc0 to $int_nets
pass in on fxp1 proto { tcp, udp } from $wwwserv to any port 53 keep state
pass out on fxp1 proto tcp from any to $wwwserv port { 21, > 49151 } flags S/SA keep state queue net_dmz_misc
pass out on fxp1 proto tcp from any to $wwwserv port 80 flags S/SA keep state queue net_dmz_http
pass out on fxp1 proto tcp from $int_nets to $wwwserv port { 80, 21, > 49151 } flags S/SA keep state queue
internal_dmz
```

FreeBSD®

# Skrypty uruchomieniowe

Plik Edycja Widok Terminal Zakładki Pomoc

```
#!/bin/sh
#
# An rc.subr-style startup script for Courier-IMAP's IMAP over SSL service.

# PROVIDE: courier_imap_imapd_ssl
# REQUIRE: LOGIN courier_authdaemond
# KEYWORD: shutdown

# Define these courier_imap_imapd_ssl_* variables in one of these files:
#       /etc/rc.conf
#       /etc/rc.conf.local
#       /etc/rc.conf.d/courier_imap_imapd_ssl
#
# DO NOT CHANGE THESE DEFAULT VALUES HERE

courier_imap_imapd_ssl_enable=${courier_imap_imapd_ssl_enable-"NO"}

. /etc/rc.subr

name="courier_imap_imapd_ssl"
rcvar=`set_rcvar`
command="/usr/local/libexec/courier-imap/imapd-ssl.rc"
pidfile="/var/run/imapd-ssl.pid"
procname="/usr/local/sbin/courierlogger"

start_cmd="imapd_ssl_cmd start"
stop_cmd="imapd_ssl_cmd stop"
restart_cmd="imapd_ssl_cmd stop && imapd_ssl_cmd start"

load_rc_config $name

imapd_ssl_cmd () {
        case $1 in
        start)
                echo "Starting ${name}."
                ${command} start
                ;;
        stop)
                echo "Stopping ${name}."
                ${command} stop
                ;;
        esac
}

run_rc_command "$1"

^? nie jest poleceniem vi
```

- Podział na systemowe i dostawców trzecich
- /etc/rc.d/
- /usr/local/etc/rc.d/
-

# FreeBSD - czyli nie taki diabeł straszny cz. 2

FreeBSD®

Mariusz Gliwiński

TLUG - Zimowisko 12.01.2008