

Multi Router Traffic Grapher

*narzędzie do monitorowania i graficznego prezentowania danych o systemach komputerowych
(w oparciu o systemy linii Windows XP i pochodne)*



Opracowanie powstało w oparciu o system operacyjny linii Windows XP SP3.
Testy wykonywane zostały na obrazie maszyny wirtualnej, który przygotowano specjalnie na potrzeby opracowania.
Wszelkie rozbieżności w adresach paczek z oprogramowaniem wynikają z ich ciągłego rozwoju.

Autorem opracowania jest *Marcin Kłopotcki*, wszelkie dane kontaktowe znajdują się na witrynie internetowej <http://mklopocki.biz>

1. Zanim zaczniesz!

Aby zapoznać się z oprogramowaniem MRTG należy wejść w posiadanie odpowiednich paczek z narzędziami.

Aby pobrać:

- x pakiet MRTG, należy udać się pod adres:
<http://oss.oetiker.ch/mrtg/pub/>
(testy wykonywano na wersji stabilnej 2.16.3)
- x pakiet Strawberry Perl, należy udać się pod adres:
<http://strawberryperl.com/>
(w opracowaniu użyto wersji stabilnej 5.12.0.1)
- x aplikację FreeSNMP, należy udać się pod adres:
<http://www.softpedia.com/get/Network-Tools/Network-IP-Scanner/FreeSnmp.shtml>
(w opracowaniu użyto wersji 1.4)



Do kompletu przyda się także nośnik instalacyjny systemu Windows, bowiem wystąpi konieczność dodania do systemu kilku istotnych bibliotek.

2. Zaraz, zaraz. Ale co to w ogóle jest?

Teraz mogę zacząć od strony teoretycznej.

Cytując Pana Artura Kulda:

MRTG (*Multi Router Traffic Grapher*) jest narzędziem służącym do monitorowania i wizualizacji niemalże dowolnych wielkości związanych z działaniem systemu komputerowego, począwszy od ruchu sieciowego (...), poprzez obciążenie procesora, pamięci, zajętość dysku i wiele, wiele innych. Istnieje możliwość dostarczenia dowolnych wielkości, o ile je tylko dostarczymy w zjadliwym przez MRTG formacie. MRTG generuje stronę HTML zawierającą wykresy w formacie .png, które dostarczają na żywo graficzne wykresy ruchu jaki odbywa się na naszych interfejsach. Używa on w dużej mierze przenośnej implementacji SNMP napisanej w całości w Perlu, dlatego też potrzebujemy jakiegokolwiek zewnętrznej paczki SNMP.

SNMP – brzmi ładnie. Ale dalej nie powiedziałem co to właściwie jest.

Simple Network Management Protocol (ang. *Prosty Protokół Zarządzania Siecią*) — standard protokołu używanego do nadzoru i zarządzania różnymi elementami sieci telekomunikacyjnych, takimi jak routery, przełączniki, komputery czy centrale telefoniczne.

cytowane za pl.wikipedia.org

Wszystko jest takie proste. Czas zabierać się za instalację.

3. Konfiguracja środowiska Windows

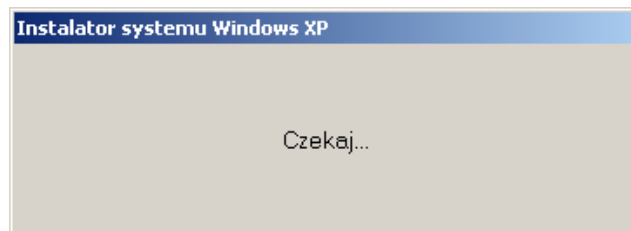
Teraz mogę się już odwoływać do definicji SNMP i założyć, że za interesowi sami zgłębią Swoją wiedzę. Otóż nasz Windows w stanie surowym nie posiada w/w usługi – jest więc bezużyteczny. Do czasu :)

Wykonaj następujące operacje:

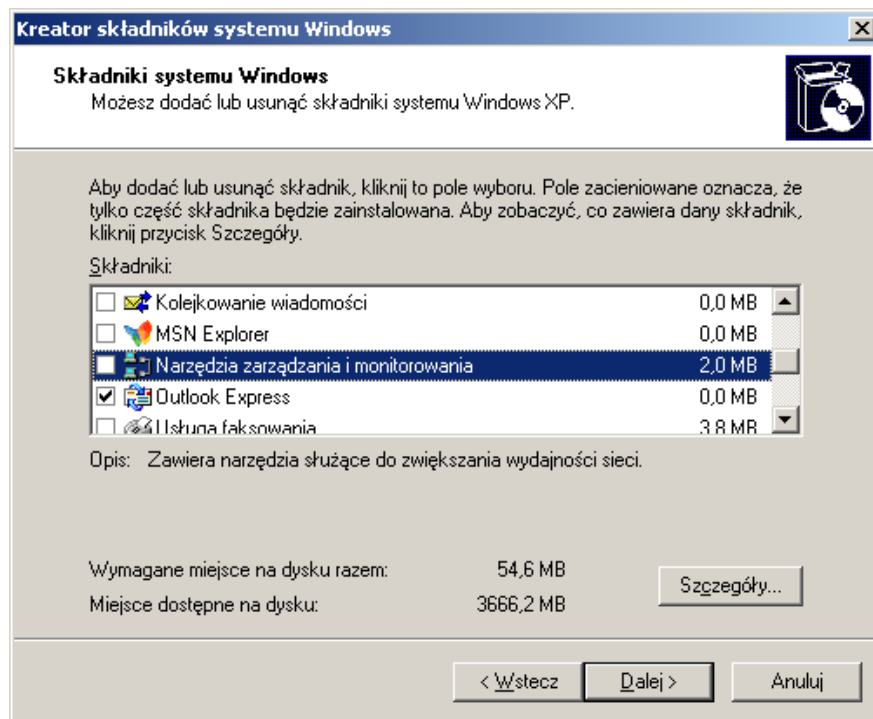
Start > Ustawienia > Panel Sterowania > Dodaj lub usuń programy

Z nowo otwartego okna wybierz polecenie:

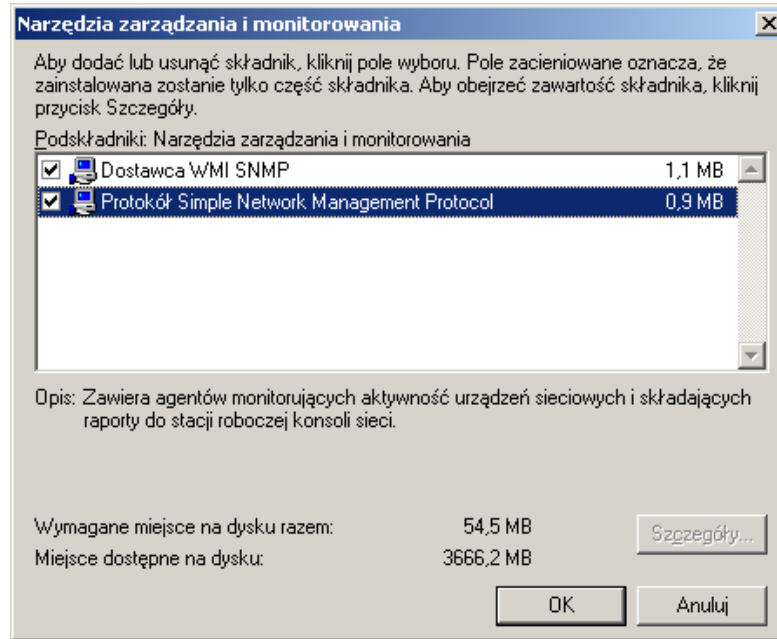
Dodaj/Usuń składniki systemu Windows



rysunek 1. Zgodnie z sugestią warto zaczekać chwilę na indeksację składników.

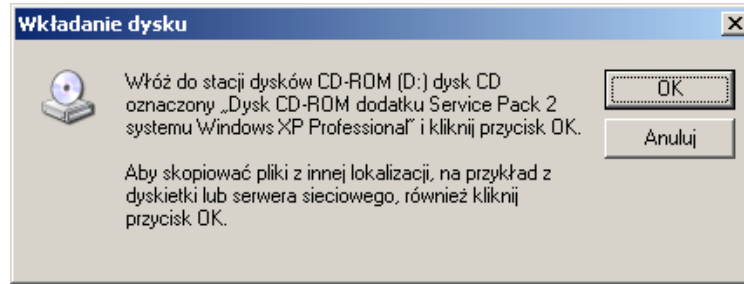


rysunek 2. Przechodzimy do „Narzędzi zarządzania i monitorowania”.

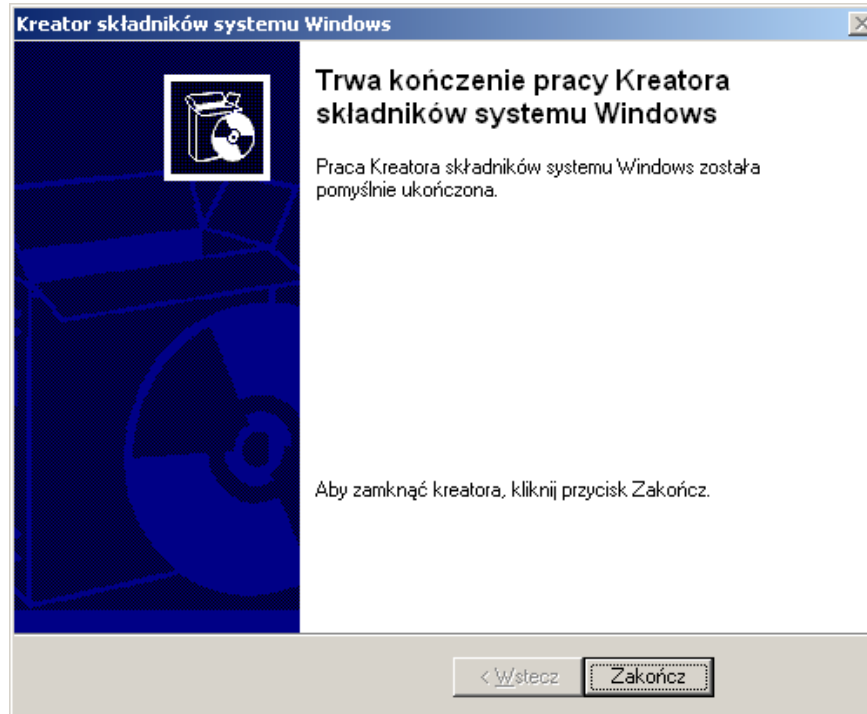


rysunek 3. Zaznaczamy obie dostępne usługi.

Jak przedstawiono na powyższych zrzutach ekranu, postępowanie sprowadza się do kilku kliknięć. Zakładając, że nośnik z systemem operacyjnym jest w napędzie procedura skończy się bez żadnego komunikatu. Jeśli nie, zostaniemy poproszeni o dostarczenie takiego nośnika do napędu z którego system został zainstalowany.



rysunek 4. Typowy błąd wynikający z pośpiechu. Dostarcz nośnik swojego systemu operacyjnego.

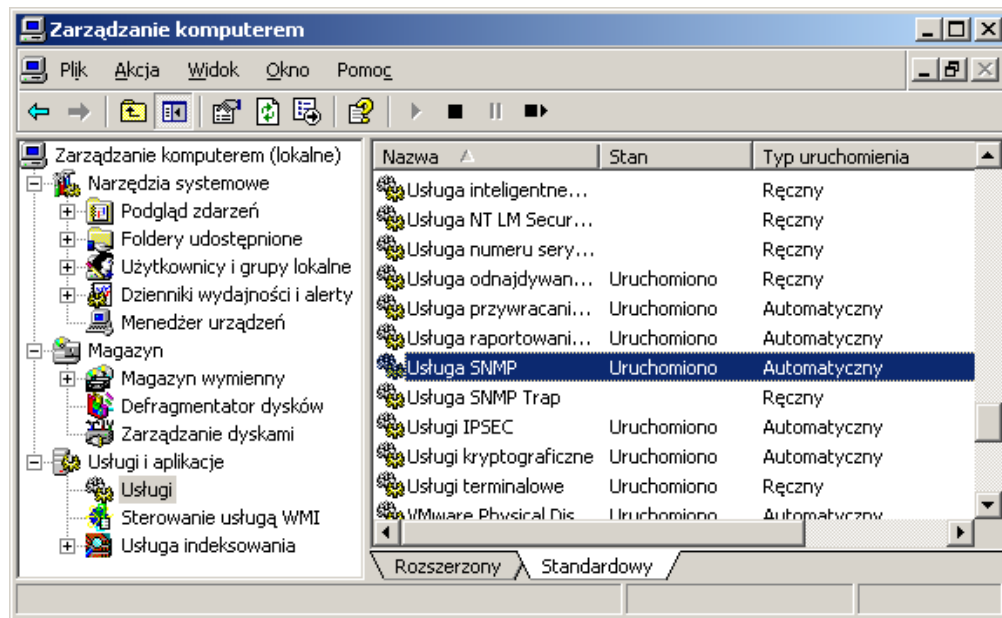


rysunek 5. Zakończono instalację składników. Wszystko przebiegło bez problemów.

Usługa SNMP dodana do listy usług systemowych? Warto upewnić się czy faktycznie dodano ją poprawnie.

Zamykamy wszystkie okna, wracamy do pulpitu i wybieramy kolejno:

Mój komputer (prawym klawiszem myszy) > Zarządzaj > Usługi i aplikacje > Usługi



rysunek 6. SNMP widnieje na liście usług.

Przystępujemy do właściwej części konfiguracji usługi.

Z wybranej na liście „Usługi SNMP”:

Usługa SNMP (prawym klawiszem myszy) > **Właściwości**

Należy sprawdzić czy usługa:

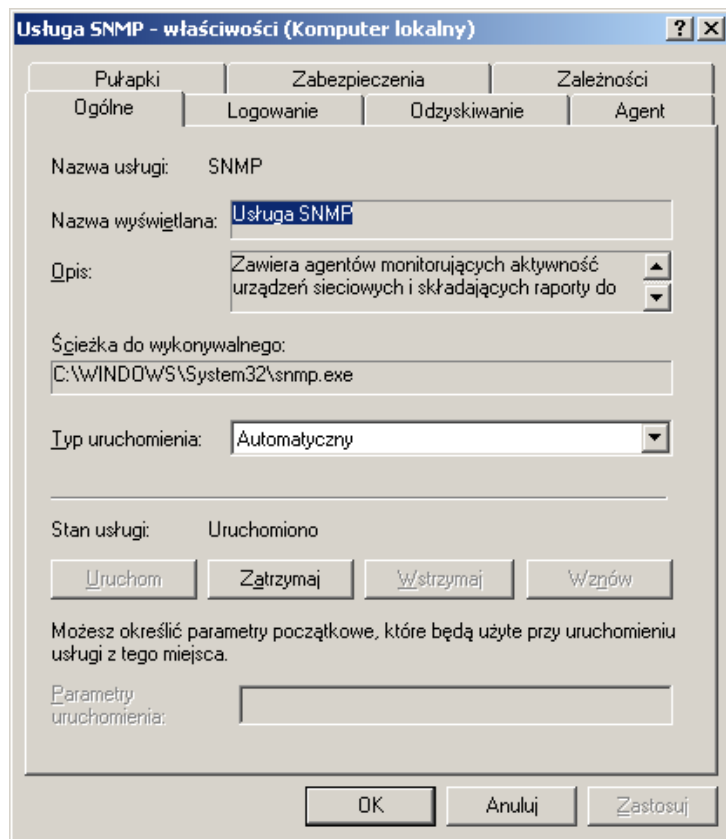
- x posiada **typ uruchomienia** ustawiony na:
Automatyczny,
- x jest włączona, jeśli nie to należy ją włączyć.

Następnie przechodzimy do zakładki **Pułapki** (rysunek 8) i ustawiamy

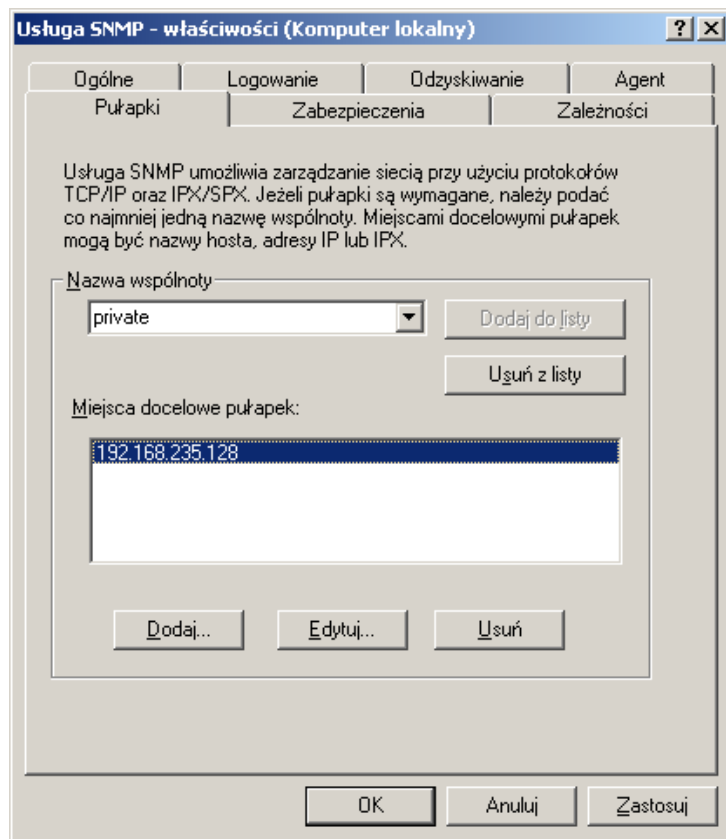
- x nazwa wspólnoty (po czym klikamy *Dodaj do listy*):
private,
- x miejsca docelowe pułapek (klikamy *Dalej*, w nowym oknie podajemy IP i zatwierdzamy poprzez *Dodaj*):
[adres IP Twojej stacji roboczej].

Następnie w zakładce **Zabezpieczenia** (rysunek 9) ustawiamy:

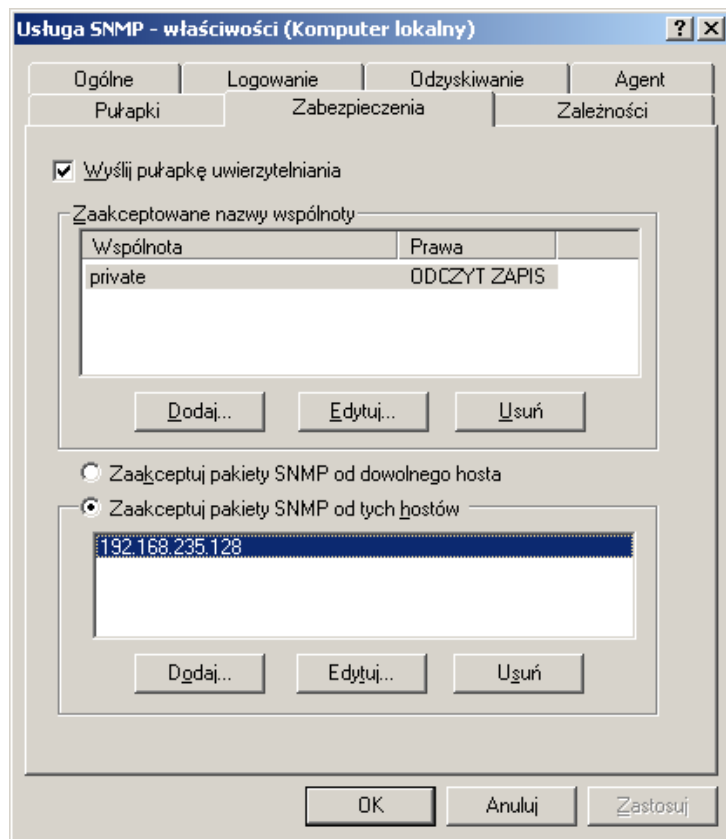
- x zaakceptowane nazwy wspólnoty:
private na **ODCZYT I ZAPIS,**
- x zaakceptuj pakiety SNMP od tych hostów:
[adres IP Twojej stacji roboczej].



rysunek 7. Właściwości ogólne usługi.



rysunek 8. Konfiguracja pułapki dla usługi.



rysunek 9. Konfigurowanie zabezpieczeń usługi.

W całym opracowaniu przewijać będą się domyślnie ustawione przeze mnie dane.

I tak:

- ✓ adres IP będzie reprezentować:
192.168.235.128 (IPv4),
- ✓ nazwa wspólnoty:
private.

Chodzi o to, aby nazwa wspólnoty zgadzała się w obu oknach, tak samo adres IP.

Dokonując takiej konfiguracji jak wyżej można zaobserwować, że chronimy się nijako przed dostępem osób z zewnątrz.

Z uwagi na prostotę metody komunikacji nie jest ona szyfrowana, ani też zabezpieczona hasłem.

Stąd wniosek aby zastosować nazwę wspólnoty dość jednoznaczną dla nas, a niekoniecznie mówiącą zbyt wiele osobom z zewnątrz.

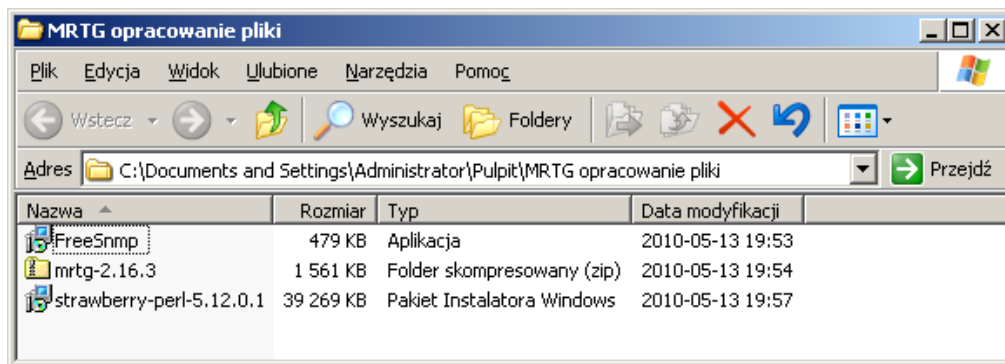
Po poprawnym ustawieniu usługi, możemy póki co o niej zapomnieć.

Zamykamy wszelkie okna i wracamy do pulpitu.

4. Pora na ... MRTG

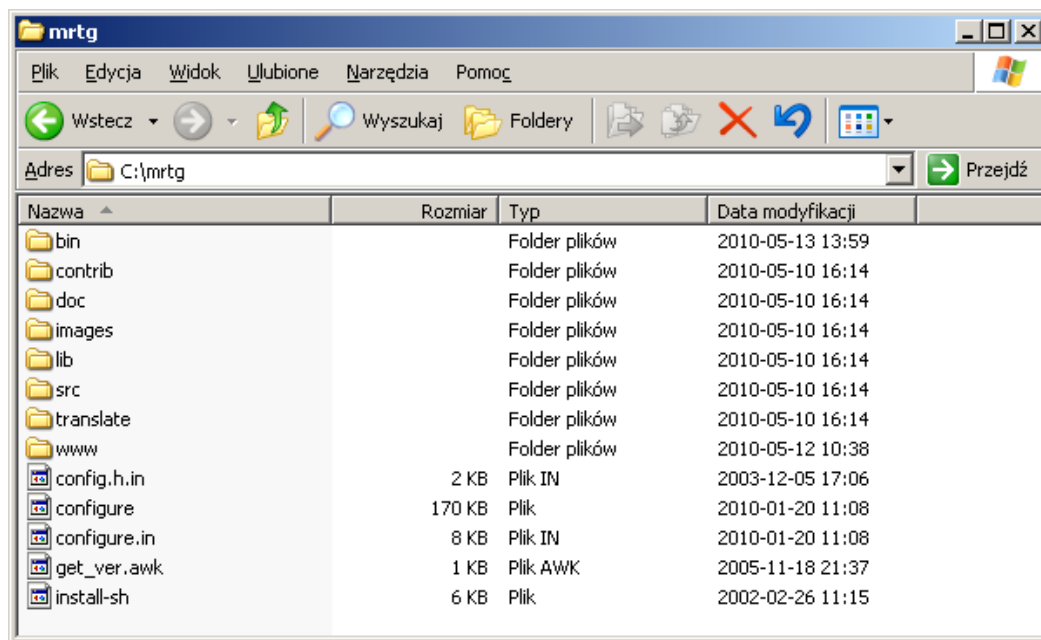
Mając folder z wymienionymi na samym początku plikami (rysunek 10) zabieramy się za instalowanie najważniejszej dla nas aplikacji.

W zasadzie słowo „instalowanie” jest zbyt na wyrost. Jest to zestaw skryptów języka PERL, które zwyczajnie należy wypakować.



rysunek 10. Folder ze wspomnianymi wcześniej plikami.

Przystępujemy do wypakowania paczki „mrtg-2.16.3.zip” do głównego folderu dysku (rysunek 11).



rysunek 11. Folder z aplikacją MRTG.

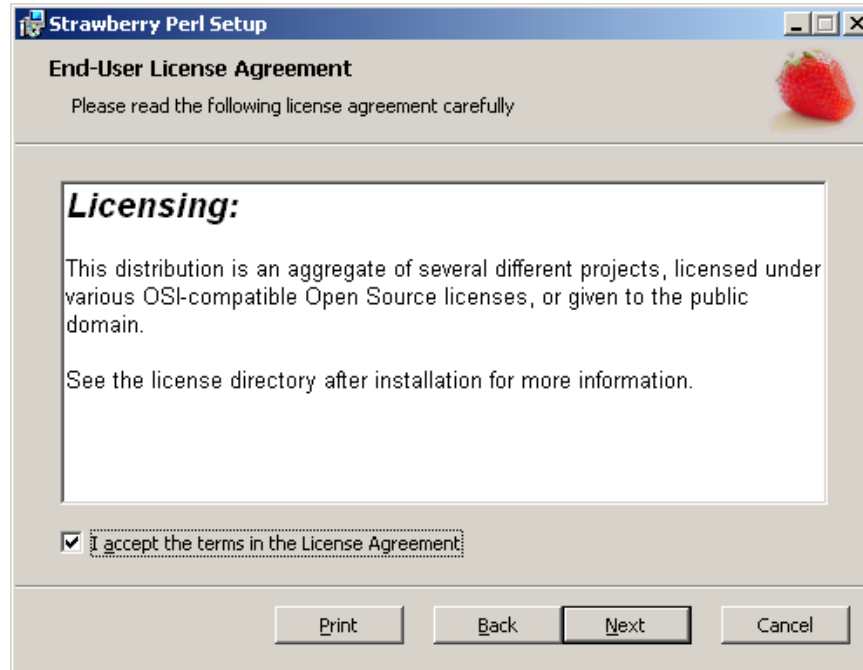
Zalecane jest aby folder aplikacji nazwać zwyczajnie „mrtg” (usuwając numer wersji), jak także wstawić w głównym katalogu dysku. Ma to na celu ukrócenie oraz uogólnienie ścieżek dostępu do aplikacji w plikach konfiguracyjnych.

5. Teraz czas na PERL'a

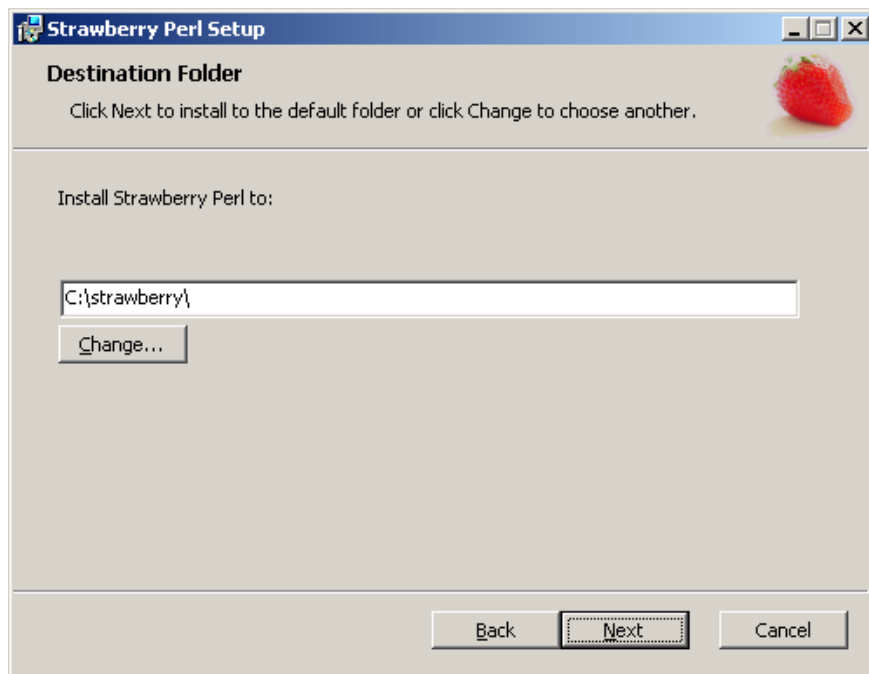
Proces instalacji tego interpretera skryptów to instalacja nazywa dość często „dalej, dalej, dalej, zakończ”.



rysunek 12. Okno powitalne instalatora Strawberry PERL.



rysunek 13. Okno umowy licencyjnej.



rysunek 14. Docelowy folder instalacji. Polecam pozostawić tak jak jest.

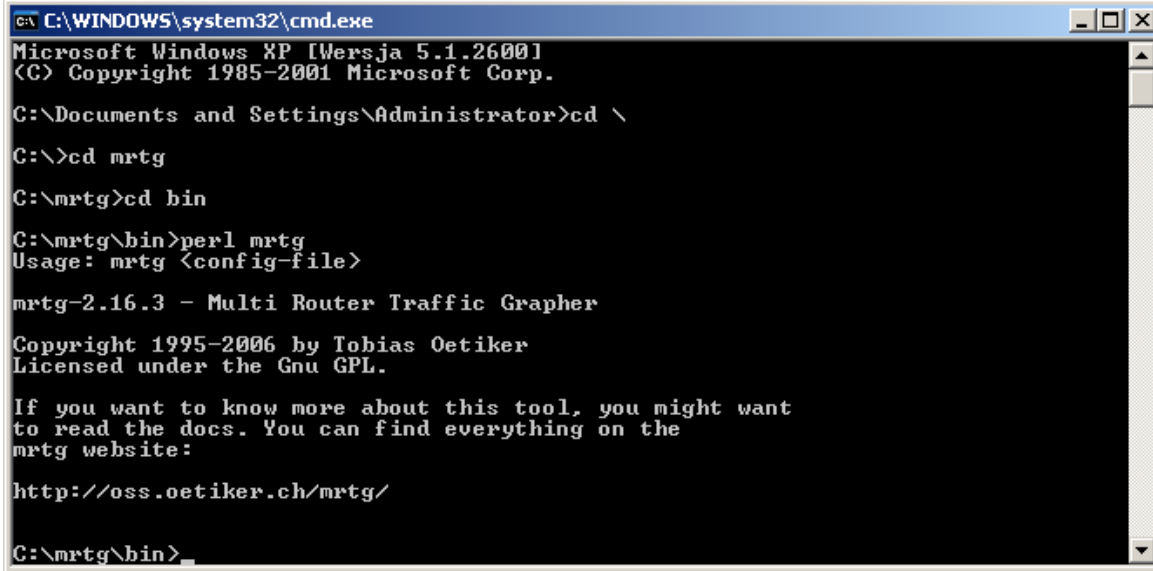
Dalej już tylko **Install** i **Finish**, więc nie będę przedstawiać tego graficznie.

6. Ale czy to działa?

Teraz zaczynamy testy naszej nowej zabawki.

Dla niektórych ta część będzie magiczna. Przystępujemy do pracy w konsoli.

Start > Uruchoń > „cmd”



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Wersja 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd \

C:\>cd mrtg

C:\mrtg>cd bin

C:\mrtg\bin>perl mrtg
Usage: mrtg <config-file>

mrtg-2.16.3 - Multi Router Traffic Grapher

Copyright 1995-2006 by Tobias Oetiker
Licensed under the Gnu GPL.

If you want to know more about this tool, you might want
to read the docs. You can find everything on the
mrtg website:

http://oss.oetiker.ch/mrtg/

C:\mrtg\bin>
```

rysunek 15. Pierwsze uruchomienie pakietu.

W ramach wprowadzenia „krok po kroku” wytłumaczę co nagle wyskoczyło nam na ekranie.

Z początkowego katalogu

```
C:\Documents and Settings\Administrator
```

powróciłem komendą

```
cd \
```

do głównego katalogu dysku.

Następnie przeniósłem się do katalogu projektu

```
cd mrtg  
cd bin
```

Ze względu na początkowe przeznaczenie bibliotek pod systemy Unix, zachowano strukturę katalogów zgodną z Unixem.

I tak mamy katalogi:

- ✓ **bin** – wersje binarne aplikacji,
- ✓ **doc** – dokumentacja,
- ✓ **lib** – dodatkowe biblioteki,
- ✓ **src** – kod źródłowy

oraz pliki pochodne.

Aplikacja uruchomiła się, choć zakończyła się bez wykonywania żadnych operacji.
No tak wymagane jest podanie pliku konfiguracyjnego, zatem do dzieła.

Ale zaraz, skąd taki plik konfiguracyjny wziąć?

Tu z pomocą przychodzi tzw. *cfgmaker*, który sam zbuduje nam początkową konfigurację.

Jak go uruchomić?

```
perl cfgmaker public@10.10.10.1 --global "WorkDir: c:\www\mrtg" --output mrtg.cfg
```

Krok po kroku co jest uruchamiane:

- ✓ silnik perla,
- ✓ *cfgmaker*,
- ✓ nazwa wspólnoty@adres IP (chyba jakiś czas temu to ustawialiśmy),
- ✓ przełącznik global ustawia jeden z globalnych parametrów konfiguracji,
 - ✓ WorkDir jest katalogiem gdzie przechowywane są pliki *html* oraz *png* z wynikowymi danymi, oraz bazy danych,
- ✓ przełącznik output wskazuje na nazwę pliku, do którego zapiszemy naszą konfigurację.

Ja uruchomiłem go tak:

```
perl cfgmaker private@192.168.235.128 --global "WorkDir: c:\mrtg\test" --output test.conf
```

Moduł aplikacji nie zgłosił błędów, za to w katalogu

```
c:\mrtg\bin
```

pojawił się nowy plik o nazwie, którą przed chwilą zdefiniowałem.

W stworzonym pliku konfiguracyjnym, który początkowo może przerażać znalazło się kilka ciekawych linii kodu.

I tak w naszej predefiniowanej konfiguracji zostały wyszukane oraz ustawione interfejsy sieciowe.

Jeden z nich to podstawowy interfejs oznaczany jako „MS-TCP-Loopback-interface”, który domyślnie jest zakomentowany.

Drugi z nich to interfejs sieciowy pod postacią karty LAN.

(z uwagi, że testy przeprowadzam w środowisku wirtualnym, stąd nazwy interfejsów nie odzwierciedlają fizycznego sprzętu w maszynie na której pracuje).

Na następnej stronie, aby pokazać całość używanej konfiguracji zamieszczam zrzut użytecznej części w/w pliku.


```
# Created by
# cfgmaker private@192.168.235.128 --global "WorkDir: c:\mrtg\test" --output test.conf

Target[192.168.235.128_2]: 2:private@192.168.235.128:
SetEnv[192.168.235.128_2]: MRTG_INT_IP="192.168.235.128" MRTG_INT_DESCR="Karta-AMD-PCNET-Family-PCI-
Ethernet---Sterownik-miniport-Harmonogramu-pakietów"
MaxBytes[192.168.235.128_2]: 125000000
Title[192.168.235.128_2]: Traffic Analysis for 2 -- MKLOPOCKI-WM
PageTop[192.168.235.128_2]: <h1>Traffic Analysis for 2 -- MKLOPOCKI-WM</h1>
    <div id="sysdetails">
        <table>
            <tr><td>System:</td><td>MKLOPOCKI-WM in </td></tr>
            <tr><td>Maintainer:</td><td></td></tr>
            <tr><td>Description:</td><td>Karta-AMD-PCNET-Family-PCI-Ethernet---Sterownik-
miniport-Harmonogramu-pakietów </td></tr>
            <tr><td>ifType:</td><td>ethernetCsmacd (6)</td></tr>
            <tr><td>ifName:</td><td></td></tr>
            <tr><td>Max Speed:</td><td>125.0 MBytes/s</td></tr>
            <tr><td>Ip:</td><td>192.168.235.128 (mklopocki-wm.localdomain)</td></tr>
        </table>
    </div>

WorkDir: c:\mrtg\test
```

Jest tu trochę zmiennych, trochę kodu html. Bałagan, ale twórczy, co mam nadzieję, że za chwilę wyjaśnię.

I tak śledząc plik konfiguracyjny znajdujemy:

- x **nazwy w nawiasach kwadratowych** – odzwierciedlają nazwy plików, które będą tworzone w czasie pracy aplikacji,
- x **Target** – do czego odwołuje się MRTG aby uzyskać dane (w chwili obecnej z interfejs sieciowy), musi być to unikalny w skali konfiguracji identyfikator,
- x **SetEnv** – ustawia dodatkowe zmienne środowiskowe, w tym przypadku MRTG_INT_IP czyli adres docelowego interfejsu i MRTG_INT_DESCR czyli jego alias,
- x **MaxBytes** – wpływa na górny zakres wykresów,
- x **Title** – nagłówek strony html (pokazywany na barku przeglądarki),
- x **PageTop** – jak wyżej z tym, że pokazywany bezpośrednio na samej stronie z podglądem informacji,

- x **WorkDir** – katalog przechowywania wygenerowanych danych.

Podsumujmy to, co udało nam się wykonać do tej pory:

- ✓ posiadamy zainstalowane aplikacje:
 - ✓ mrtg,
 - ✓ perl,
- ✓ posiadamy podstawową konfigurację mrtg aby uruchomić system zbierania danych.

Przystąpmy do pierwszego uruchomienia naszego środowiska zbierania danych.

Powracamy do naszej konsoli, poleceniem `cd` przechodzimy do katalogu

```
c:\mrtg\bin
```

I wydajemy polecenie

```
perl mrtg test.conf
```

Aplikacja uruchomi się, stworzy pliki niezbędne do pracy i zakończy swoje działanie, bowiem tak została przez nas skonfigurowana.

Wynikiem pracy aplikacji będzie wygenerowanie zestawu plików:

- ✓ *[uniqueID].html* – będącego stroną z zawartością wszystkich wygenerowanych wykresów,
- ✓ oraz dodatkowo zestaw plików graficznych:
 - ✓ *[uniqueID]-day.png* – będący wykresem działania w ciągu ostatnich 12 godzin,
 - ✓ *[uniqueID]-week.png* – wykres tygodniowy,
 - ✓ *[uniqueID]-month.png* – wykres miesięczny,
 - ✓ *[uniqueID]-year.png* – wykres roczny.

Przy powyższych ustawieniach to my powinniśmy dbać aby aplikacja uruchamiana była w stałych odstępach czasowych.

W dalszej części wyjaśnię co należy zmienić aby MRTG pracowało samodzielnie (w tle), bez naszej ingerencji.

To w zasadzie koniec podstawowej konfiguracji narzędzia.

Działa, zbiera dane dotyczące interfejsu sieciowego – w podstawowej wersji nie oferuje zbyt rozbudowanych statystyk.

Natomiast dane, które generuje nie są dla mnie osobiście zadowalające.

Aplikacja monitoruje zbyt mało parametrów komputera, wykresy nie porywają (tzn są w monotonnym odcieniu zieleni), każdy z wykresów należy oglądać osobno (brak strony podsumowania).

Wiele można poprawić i o tym postaram się opowiedzieć w dalszej części.

Wykresy użycia procesora



Witam w drugiej części artykułu o **MRTG**, systemie monitorowania danych dotyczących komputera.

W tej części pokrótce omówię jak dołączyć wykres użycia procesora do całej paczki monitorowanych danych.

Z moich obserwacji wynika, że MRTG uruchamiane na systemie operacyjnym Linux i pochodnych otrzymuje dane w sposób inny, niż założyłem że zaprezentuję.

W większości tutoriali znalezionych w sieci Internet użytkownicy jasno wskazują, że dane do statystyk dostarczają poprzez zewnętrzne skrypty. Chcąc tego uniknąć skorzystamy z wcześniej zainstalowanej usługi SNMP.

Wskazemy skąd pobierać dane, jak je ewentualnie przetwarzać i na ich podstawie generować dynamiczne wykresy.

Na początek statystyki użycia CPU.

Na następnej stronie przykład pliku konfiguracyjnego wraz ze szczegółowym omówieniem.

```
Target[cpu]:      ( .1.3.6.1.2.1.25.3.3.1.2.1&.1.3.6.1.2.1.25.3.3.1.2.1:private@192.168.235.128 ) / 1
Title[cpu]:      VMware - uzycie CPU
PageTop[cpu]:    <H1>VMware - uzycie CPU</H1>
MaxBytes[cpu]:   100
ShortLegend[cpu]: %
YLegend[cpu]:    wartosc (%)
Legend1[cpu]:    procentowe uzycie CPU
LegendI[cpu]:    Uzyte
Legend0[cpu]:
Options[cpu]:    growright,nopercent,gauge
Unscaled[cpu]:   ymwd
```

Ogólny schemat pliku konfiguracyjnego podałem powyżej, ale przytoczę go raz jeszcze dla lepszego zrozumienia do czego przydatne są poszczególne linijki.

Tytuł strony HTML (nazwa wyświetlana na barku przeglądarki):

```
Title[cpu]:      VMware - uzycie CPU
```

Nagłówek na stronie z podglądem statystyk:

```
PageTop[cpu]:    <H1>VMware - uzycie CPU</H1>
```

Maksymalna wartość danych na wykresach:

```
MaxBytes[cpu]:   100
```

Skrócony opis legendy (występujący tuż pod opisami wykresów):

```
ShortLegend[cpu]: %
```

Opis osi Y na wykresie (gdy brak, wartość domyślnie ustawiana przez system):

```
YLegend[cpu]:      wartosc (%)
```

Legenda dla pierwszego koloru wykresu: (szczegółowo o kolorach powiem po omówieniu sekcji monitorujących)

```
Legend1[cpu]:      procentowe uzycie CPU
```

Legenda dla ruchu przychodzącego:

```
LegendI[cpu]:      Uzyte
```

Legenda dla ruchu wychodzącego:

```
Legend0[cpu]:
```

Dodatkowe opcje przy generowaniu wykresów:

```
Options[cpu]:      growright,nopercent,gauge
```

Skalowanie wykresów na zbiorczej podstronie informacyjnej (y-rok, m-miesiąc, w-tydzień, d-dzień):

```
Unscaled[cpu]:    ymwd
```

Na koniec pozostała dość skomplikowana na pierwszy rzut oka linijka:

Miejsce skąd pobierać dane:

```
Target[cpu]: ( .1.3.6.1.2.1.25.3.3.1.2.1&.1.3.6.1.2.1.25.3.3.1.2.1:private@192.168.235.128 ) / 1
```

I tu zaczyna się prawdziwa przygoda z protokołem SNMP.

Składnia wywołania to:

```
OID & OID : wspólnota @ host
```

Pojawiła się kolejna definicja do wyjaśnienia.

OID (ang. *Object Identifier*) - unikatowy identyfikator obiektu, służy do odróżnienia obiektu od innych obiektów oraz do tworzenia odwołań do tego obiektu przez system. Użytkownik posługuje się nazwą obiektu, natomiast system zamienia ją na identyfikator.

cytowane za pl.wikipedia.org

I dokładnie po identyfikatorze OID protokół komunikuje się z hostem i prosi o dostarczenie danych dotyczących konkretnego urządzenia. Jest to metoda prostsza jeśli wiemy czego szukamy, jeśli zaczynamy przygodę z OID'ami to wydaje się to dość trudne i skomplikowane.

Gałąź OID:

```
.1.3.6.1.2.1.25.3.3.1.2.X
```

rozpoznawana także jako:

```
host.hrDevice.hrProcessorTable.hrProcessorEntry.hrProcessorLoad.X
```

Gdzie pod X podstawiamy numer naszego procesora. (dla jednostek jedno procesorowych: 1), co daje nam dane dotyczące wykorzystywania procesora.

Analogicznie dla systemów wieloprocesorowych będziemy się zwyczajnie odwoływać kolejnymi cyframi większymi od 1.

Z uwagi na fakt, iż musimy dostarczyć w jednym cyklu odczytu minimum dwie dane do obróbki działamy w/g schematu:

- x dwa razy odczytujemy żądaną wartość,
- x spajamy ze sobą dwa odczyty operatorem łączenia (&).

I na koniec podajemy dostęp do naszej wspólnoty i hosta, z którego odczytu chcemy dokonać.

Dokładnie to narzędzie jest szeroko używane do śledzenia parametrów urządzeń zdalnych, ale w naszym prostym przypadku poprzestaniemy na maszynie lokalnej.

Przeniesienie konfiguracji na urządzenia zdalne nie powinno nastroić kłopotów.

Wykresy użycia pamięci RAM



W tym przypadku sprawa się odrobinę komplikuje. Wynika to z faktu iż od systemu docelowego nie otrzymamy odpowiedzi na pytanie „ile pamięci jest wykorzystywanej. wynik podaj w procentach”.

Jeśli zapytamy „ile pamięci aktualnie wykorzystuję?” też nie dostaniemy odpowiedzi, bowiem pod zmienną za to odpowiedzialną jest bliżej nieokreślona wartość, która nijak ma się do pamięci RAM.

To samo czeka nas jeśli zapytamy o łączną ilość pamięci RAM. Także bliżej niezwiązana z niczym liczbą.

Natomiast jeśli obie liczby zestawimy ze sobą, dadzą bardzo ładny procentowy wynik – co najlepsze zgodny z faktycznym stanem użycia pamięci.

I podobnie jak w przypadku CPU poniżej fragment konfiguracji, z tym że bez omówienia powtarzających się wpisów.

```
Target[ram]:      ( .1.3.6.1.2.1.25.2.3.1.6.5&.1.3.6.1.2.1.25.2.3.1.6.5:private@192.168.235.128 ) /  
                  ( .1.3.6.1.2.1.25.2.3.1.5.5&.1.3.6.1.2.1.25.2.3.1.5.5:private@192.168.235.128 ) * 100  
Title[ram]:      VMware - uzycie RAM  
PageTop[ram]:    <H1>VMware - uzycie RAM</H1>  
MaxBytes[ram]:   100  
ShortLegend[ram]: %  
YLegend[ram]:    wartosc (%)  
Legend1[ram]:    procentowe uzycie RAM  
LegendI[ram]:    Uzyte  
LegendO[ram]:  
Options[ram]:    growright,nopercent,gauge,integer  
Unscaled[ram]:   ymwd
```

Gałąź OID:

```
.1.3.6.1.2.1.25.2.3.1.X.X  
rozpoznawana także jako:  
host.hrStorage.hrStorageTable.hrStorageEntry.X.X
```

odpowiada za przechowywanie danych dotyczących fizycznie spiętych z systemem urządzeń.

Jak dotrzeć do pamięci i dlaczego pamięć RAM ma przydzielony numer 5 w tablicy?

Otóż jest to bardzo proste i logiczne.

Numerowanie wynika z zasady:

- x stacje dysków miękkich (stacja dyskietek 3,5 cala, ZIP itd.),
- x stacje dysków twardych (listowane partycjami),
- x stacje dysków optycznych,
- x kontroler pamięci wirtualnej,
- x kontroler pamięci fizycznej.

Stąd w moim przypadku wygląda to tak:

- x stacja dyskietek / A /:
- x dysk twardy / C: /,
- x napęd CD / D: /,
- x Virtual Memory,
- x Physical Memory.

(inaczej niż w przypadku programowania, tablica numerowana jest od jedyнки)

Co, jak, gdzie, dlaczego? Czyli gdzie znaleźć zużycie, a gdzie dostępną ilość pamięci?

```
.1.3.6.1.2.1.25.2.3.1.5.X
```

rozpoznawana także jako:

```
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageSize.X
```

Mamy dostęp do „całkowitej dostępnej” ilości pamięci.

```
.1.3.6.1.2.1.25.2.3.1.6.X
```

rozpoznawana także jako:

```
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageUsed.X
```

Mamy dostęp do „użytej” ilości pamięci.

Jak wcześniej mówiłem, te liczby nie odzwierciedlają danych o pamięci.

Jednak wyliczenie ze wzoru:

$$\frac{\text{pamięć używana}}{\text{pamięć dostępna}} \cdot 100$$

pozwała uzyskać procentowe użycie, które na wykresie wygląda zdecydowanie przyjemniej, niż zwykłe dane liczbowe.

Czy da się to jakoś szybko policzyć?

Powróćmy do przykładu z wycinka pliku konfiguracyjnego.

```
Target[ram]:      ( .1.3.6.1.2.1.25.2.3.1.6.5&.1.3.6.1.2.1.25.2.3.1.6.5:private@192.168.235.128 ) /  
                  ( .1.3.6.1.2.1.25.2.3.1.5.5&.1.3.6.1.2.1.25.2.3.1.5.5:private@192.168.235.128 ) * 100
```

Czy tu przypadkiem nie użyłem podstawowych operatorów matematycznych?

Dokładnie, każdą daną po odebraniu możemy przetworzyć.

Także nie ma problemu z tym, że zwracamy podwójne dane. Zostaną przeliczone inteligentnie.

Nic nie będzie pomieszane, wszystko zgodnie z naszym zamysłem.

Z każdą chwilą przekonuję się, że narzędzie to jest naprawdę proste z użyciu. A Wy?

Wykresy użycia uruchomionych procesów użytkownika



Spójrzmy na przykład kodu konfiguracji:

```
Target[processes]:      .1.3.6.1.2.1.25.1.6.0&.1.3.6.1.2.1.25.1.6.0:private@192.168.235.128
Title[processes]:      VMware - uzycie procesow
PageTop[processes]:    <H1>VMware - uzycie procesow</H1>
MaxBytes[processes]:   1000
ShortLegend[processes]: procs
YLegend[processes]:    .
Legend1[processes]:    Procesy
LegendI[processes]:    Uruchomionych
LegendO[processes]:
Options[processes]:    growright,nopercent,gauge
```

Gałąź OID:

```
.1.3.6.1.2.1.25.1.6.0
rozpoznawana także jako:
host.hrSystem.hrSystemProcessess.0
```

Przechowuje ilość uruchomionych procesów, które bezpośrednio można użyć do stworzenia wykresu.

Wykresy użycia nawiązanych połączeń TCP



Spójrzmy na przykład kodu konfiguracji:

```
Target[newTcpConn]:      tcpPassiveOpens.0&tcpActiveOpens.0:private@192.168.235.128
Title[newTcpConn]:      VMware - tworzenie nowych polaczen TCP
PageTop[newTcpConn]:    <H1>VMware - tworzenie nowych polaczen TCP</H1>
MaxBytes[newTcpConn]:   100000000000
ShortLegend[newTcpConn]: pol/sek
YLegend[newTcpConn]:    .
LegendI[newTcpConn]:    Wchodzace
LegendO[newTcpConn]:    Wychodzace
Legend1[newTcpConn]:    Nowe polaczenia przychodzace
Legend2[newTcpConn]:    Nowe polaczenia wychodzace
Options[newTcpConn]:    growright,nopercent,perminute
```

Gałąź OID:

```
tcpPassiveOpens.0
tcpActiveOpens.0
```

przechowuje wartości, które bezpośrednio umieszczamy na wykresie.

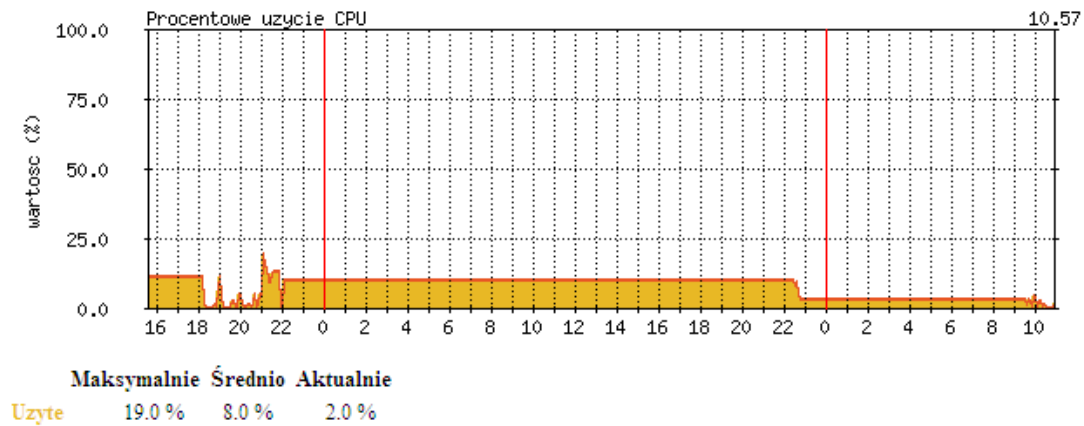
Zaraz zaraz...

Jak pewnie zauważyliście w tym przypadku przekazaliśmy dwie inne dane?
A wcześniej przekazywaliśmy cały czas te same tylko, każdą dwa razy.

Istotnie. Na jednym wykresie możemy umieścić do 4 różnych danych.
Przy czym wymogiem jest umieszczenie 2 danych, aby na ich podstawie generować wykresy.

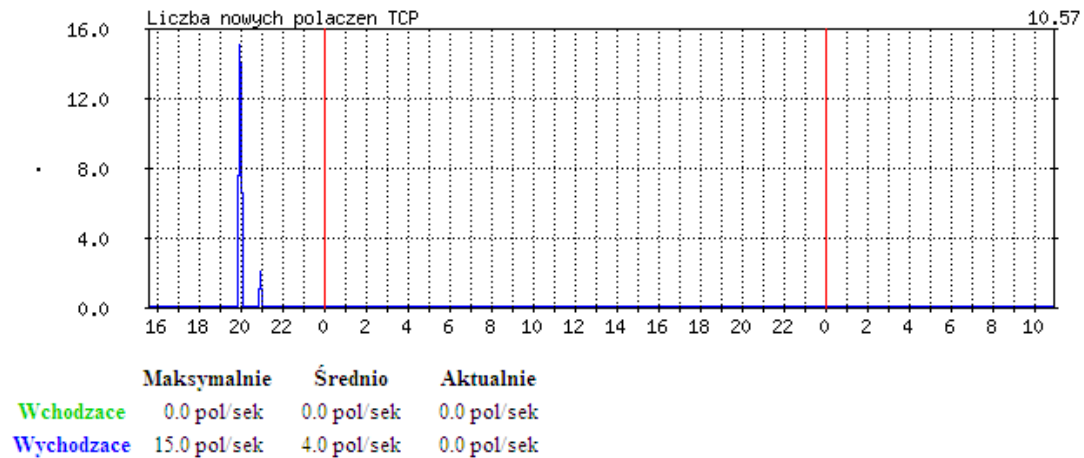
I tu pojawia się kolejna inteligentna cecha systemu. Jeśli przekażemy dwie takie same dane, system to wychwyci i przedstawi na wykresie tylko pierwszą z nich. Widać to szczegółowo w legendzie pod wykresami.

Wykres CPU (przekazane dwie takie same dane):



rysunek 16. Wykres CPU z legendą.

Wykres połączeń TCP (przekazane dwie różne dane):

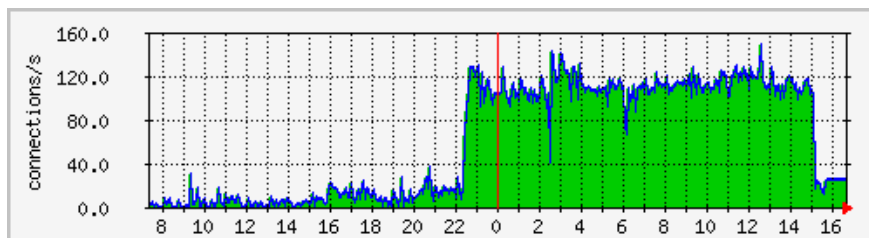


rysunek 17. Wykres liczby połączeń TCP z legendą.

Ożywiamy wykresy



Jak zapewne zwróciliście uwagę, wykresy opisane numerami 15 i 16 nie wyglądają tak zwyczajnie.



rysunek 18. Podstawowy typ wykresu: zielone pole i niebieska ramka.

Nie ma nic trudnego, żeby każdy z wykresów wyglądał inaczej.
Inaczej czyli w innych kolorach, z innym tłem i w innych rozmiarach.

Do tego stworzono zmienne konfiguracyjne:

```
Xsize[ uniqueID ]: 300  
Ysize[ uniqueID ]: 300
```

(warunek: rozmiar X musi zawierać się pomiędzy wartościami <20, 600>; rozmiar Y musi być większy niż 20)

Odradzam skalowania i wykorzystywania opcji zoom.
Dają paskudne rezultaty, stąd też nie widzę potrzeby aby mówić o nich więcej.

Zmiana kolorów? Nic prostszego.

```
Colors[ uniqueID ]:  Rozowy#FF00FF,Ciemno rozowy#800080,Trzeci#993615,Czwarty#997815
```

Wymogi?

Wymagane jest podanie 4 kolorów w zapisie heksadecymalnym wraz z ich opisami.

Zapis?

- ✓ Nazwa koloru,
- ✓ znak (#),
- ✓ heksadecymalny kod koloru RGB (bez poprzedzającego znaku (#)).

(zapis powtórzony czterokrotnie)

Jak MRTG pobiera kolory do wykresów?

- x Jeśli przekazano jedną daną (w tym podwójnie jedną daną):
 - x pierwszy kolor: wypełnienie pola pod wykresem,
 - x drugi kolor: obramowanie wykresu,
 - x trzeci i czwarty kolor: nieużywane.
- x Jeśli przekazano dwie i więcej różnych danych:
 - x w zależności od ich ilości - taka liczba kolorów,
 - x pole pod wykresem nie jest malowane, aby nie ograniczać czytelności wykresu.

Tytuł wykresu? Proszę bardzo.

```
PNGTitle[ uniqueID ]: Tytuł
```

Nie polecam używania polskich znaków, ani w tytułach ani w opisach.

Domyślne kodowanie danych to **ISO-8859-2** zatem zalecam ostrożność – błędy kodowania psują estetykę.

Życzycie sobie godzinę ostatniej aktualizacji na wykresie?

```
TimeStrPos[ uniqueID ]: RU  
TimeStrFmt[ uniqueID ]: %H.%M
```

Jest to zapis polecany.

Pierwsza linijka odpowiada za położenie godziny (Right Upper, Left Upper, Right Lower and Left Lower, bądź też NO)

Druga linia to zapis czasu na wykresie.

Do wykorzystania są wskaźniki czasu:

- ✓ %Y – rok,
- ✓ %m – miesiąc,
- ✓ %d – dzień,
- ✓ %H – godzina,
- ✓ %M – minuty,
- ✓ %S – sekundy.

Zmieniamy tło wykresu? Proste.

```
Background[ uniqueID ]: #FFFFFF
```

Zapis koloru?

Heksadecymalny z wcześniejszym znakiem (#) poprzedzającym kolor.

I na koniec tej części kilka dodatkowych opcji, które znacząco wpływają na wygląd wyresów.

```
Options[ uniqueID ]: transparent, noborder, noarrow, nobanner, nolegend
```

Za co odpowiadają poszczególne parametry?

- ✓ transparent – przezroczyste tło,
- ✓ noborder – brak ramki wokół wykresu (bardzo znacząco podnosi jakość wykresów),
- ✓ noarrow – likwiduje strzałkę na wykresie dziennym (przy aktualnej godzinie),
- ✓ nobanner – usuwa banner MRTG ze stopki strony ze statystykami,
- ✓ nolegend – usuwa legendę podsumowującą wygenerowane statystyki (ze stopki strony).

Kilka trików podczas tworzenia konfiguracji.

Wpisywanie tych samych linijek do każdej z sekcji konfiguracji jest nudne, żmudne i pozbawione sensu? Zgadzam się.

Stąd istnieją dwa triki aby ułatwić sobie życie.

Parametr globalny:

```
ParameterName[ _ ]: value
```

Wpisuje globalne ustawienie dla każdej sekcji.

Dobre do ustawienia wielkości wykresów, tła, ustawienia czasu.

Parametr wymagalności:

```
ParameterName[ ^ ]: value
```

Moglibyśmy ustawić parametr globalny, ale jeśli go nadpiszemy – początkowe dane **zostaną utraczone**.

Natomiast ustawiając parametr wymagalności dane ustawione w sekcji zostaną **połączone** z danymi wymaganymi.

Paczki językowe



Pakiet MRTG posiada wkompiłowaną obsługę języka polskiego.
Jego ustawienie to cała jedna linijka.

```
Language: polish
```

Jednak dla mnie ten „langpack” jest trochę zbyt „wydumany”.
Co mam na myśli – niektóre sekcje są tłumaczone bardzo dokładnie z angielskiego, przez co brzmią bardzo technicznie.
Mówiąc szczerze nie podoba mi się to tłumaczenie.

Oczywiście podziękowania dla Łukasza Jokiela za wkład w przetłumaczenie pakietu.
Oczywiście zostawiamy wszystkie informacje o autorze. A w pliku dokonujemy kilku korekt.
Co zmienicie – zależy od Was.

Jak teraz wprowadzić w życie nasze zmiany w pliku językowym?
Zapisać i zostawić? Niestety. Translacja wymaga kompilacji na format przyjazny bibliotekom MRTG.
Ale aby ułatwić nam życie wystarczy użyć odpowiedniego narzędzia.

Przy użyciu konsoli przemieszczamy się do katalogu:

```
c:\mrtg\translate
```

Wykonujemy następujące operacje:

```
perl mergeLocate.pl polish.pmd german.pmd
```

Skrypt nie zwróci żadnego komunikatu natomiast w pliku:

```
c:\mrtg\translate\locales_mrtg.pm
```

znajduje się wersja gotowa do użycia przez biblioteki językowe.

Wystarczy skopiować ją do katalogu :

```
c:\mrtg\lib\mrtg2\
```

i wydać zgodę na nadpisanie istniejącego tam pliku.

Dlaczego kompilujemy pliki „*polish.pmd*” i „*german.pmd*”? Sama polska lokalizacja nie wystarczy? Niestety, przy jednej lokalizacji system MRTG nie chce wystartować. Przy dwóch problem rozwiązuje się sam.



MRTG jako usługa systemowa



W chwili obecnej nasza konfiguracja tworzy bazę danych, wykresy oraz strony ze statystykami „na żądanie”.

Jest to uciążliwe bowiem wymusza na nas pilnowanie i uruchamianie aplikacji co ścisły okres czasu.
Czy nie można tego zautomatyzować?

Można!

Potrzebna jest dodatkowa paczka plików:

- ◆ zestaw plików przygotowany specjalnie na potrzeby tego opracowania <http://trash.klocu.info/mrtg/service.zip>.

Pliki wypakowujemy do folderu:

```
c:\mrtg\bin\
```

tak aby lista plików zawierała pliki:

- ✓ instsrv.exe,
- ✓ srvany.exe,
- ✓ regedit_service.reg.

Do pliku konfiguracyjnego dopisujemy linijki:

```
RunAsDaemon: yes  
EnableIPv6: no  
Interval: 5
```

Korzystamy z nowych ustawień globalnych:

Do pliku konfiguracyjnego dopisujemy linijki:

```
RunAsDaemon: yes
```

Uruchamiamy MRTG jako demon systemowego. Jest to usługa siedząca w pamięci i będąca w stanie uruchomionym przez cały czas pracy systemu operacyjnego.

```
EnableIPv6: no
```

Wyłączamy korzystanie z opcji Ipv6.

Jeśli nie posiadamy takowej adresacji – nie wpłynie to znacząco na działanie usługi.

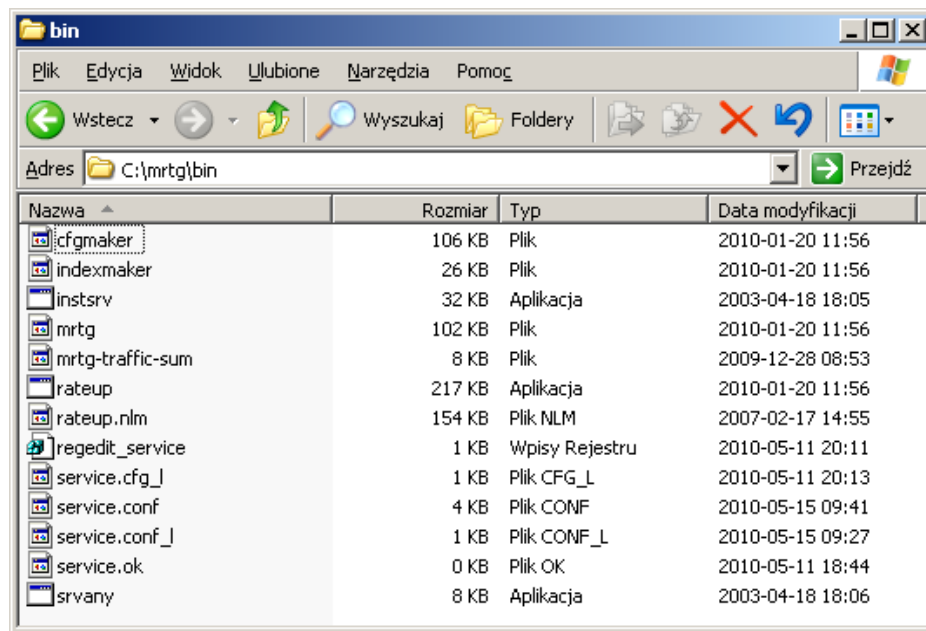
Jeśli posiadamy adres Ipv6 i wyłączymy jego używanie, przyspieszymy delikatnie pracę usługi.

Dlaczego? Bowiem na początku poszukiwana jest nazwa hosta po adresie IPv6, w przypadku niepowodzenia wyszukiwanie wykonywane jest poprzez IPv4. Czyli likwidujemy jeden cykl pracy na starcie.

```
Interval: 5
```

Częstotliwość odświeżeń statystyk. Minimalny czas dla MRTG to 5 minut. Istnieje patch modyfikujący pliki biblioteki, tak aby uzyskać częstsze wyzwalanie aktualizacji.

Wspomniany patch został wydany dla dość wiekowej (obecnie) wersji stąd nie stosowałem go w czasie pracy z biblioteką.



rysunek 19. Folder plików binarnych i konfiguracyjnych MRTG.

Plik „*regedit_service.reg*” powinien zawierać wpisy:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MRTG\Parameters]
"Application" = "c:\\strawberry\\perl\\bin\\perl.exe"
"AppParameters" = "c:\\mrtg\\bin\\mrtg c:\\mrtg\\bin\\service.cfg"
"AppDirectory" = "c:\\mrtg\\bin\\"
```

Plik integrujemy z rejestrem systemu:

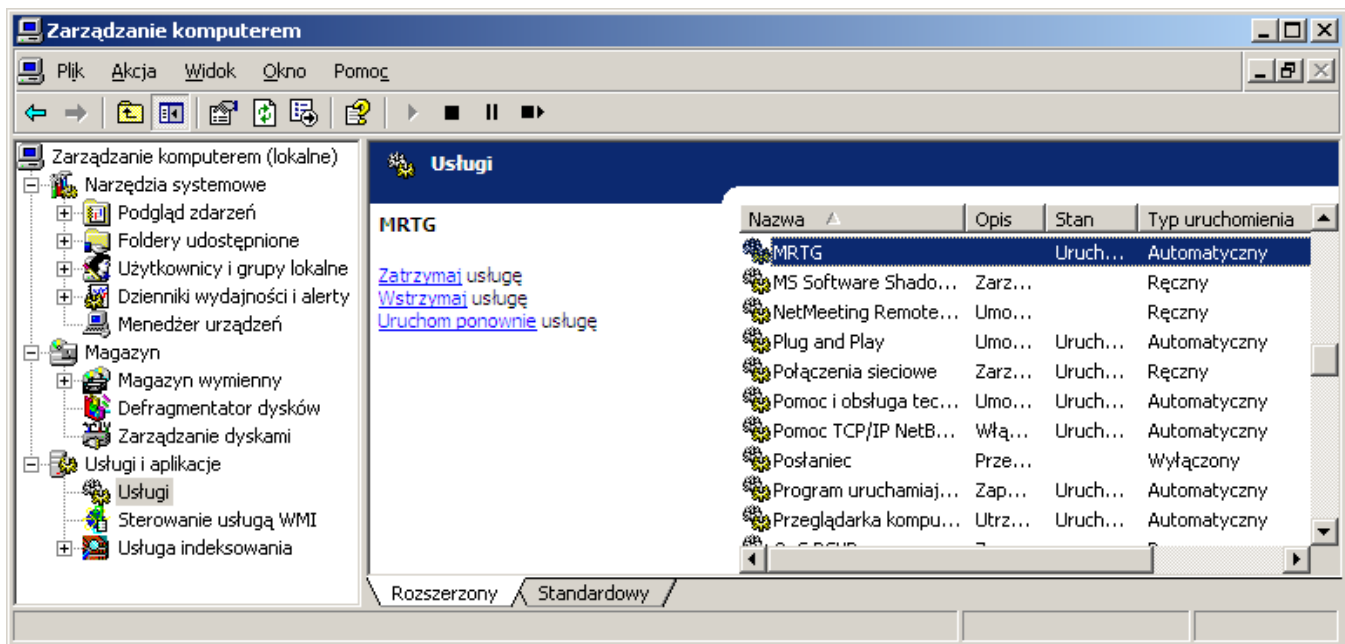
- ✓ klikamy prawym klawiszem na plik i wybieramy **Scał**,
- ✓ klikamy dwukrotnie na plik: potwierdzamy zgodę na integrację pliku z rejestrem, po czym zamykamy okno potwierdzenia.

Doda to usługę systemu Windows do listy usług systemowych (rysunek 20).
(nie musimy dodawać, że należy uwzględnić własne ścieżki do plików!)

Co jest ważne, usługa ma wszystkie opcje:

- x Uruchom,
- x Zatrzymaj,
- x Wstrzymaj,
- x Uruchom ponownie.

Nie polecam korzystać z opcji „Wstrzymaj”.



rysunek 20. MRTG jako usługa systemu Windows.

Czy istnieje ryzyko potencjalnej awarii.

„Uruchomiłem usługę, ale nie aktualizuje ona plików. Mam stare wykresy i pliki.”

Usługa nie zwróciła kodu błędu . Dlaczego? Proszę zwrócić uwagę na sposób uruchamiania.

Główna aplikacja to **silnik Perla**.

Następnie uruchamiamy **MRTG** i jako parametr przekazujemy mu adres pliku konfiguracyjnego.

Znakiem tego Perl nie przekazuje dalej kodów błędów, a jedynie zakańcza wykonywanie skryptu w momencie jeśli ten odnajdzie błąd w plikach konfiguracyjnych.

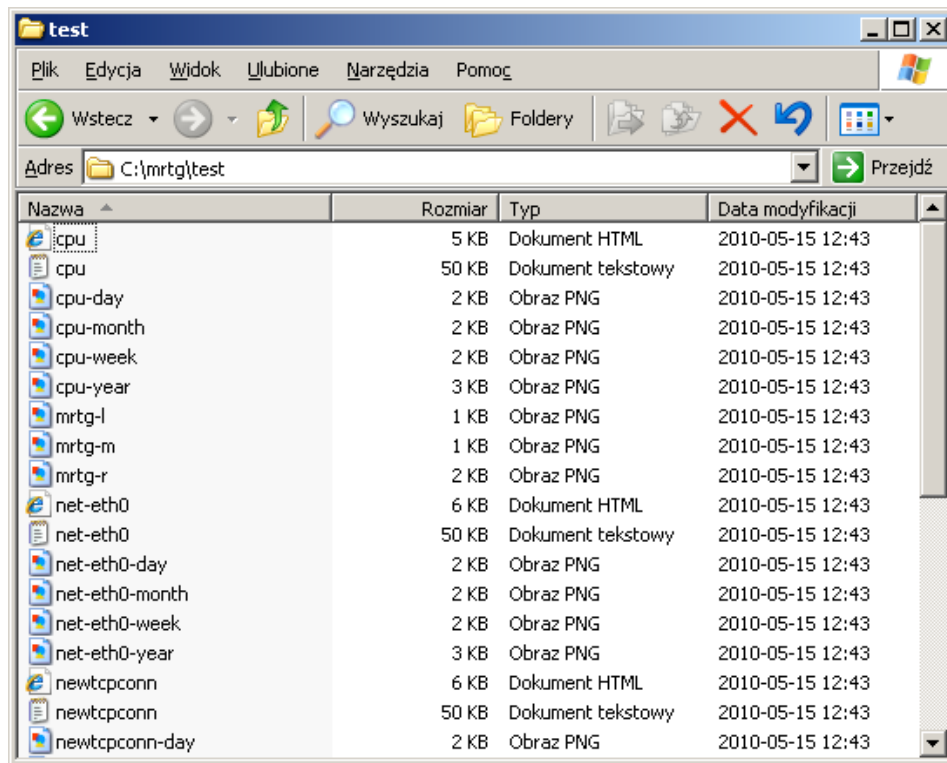
Moja rada: **Zatrzymać usługę** poprzez panel **Usług**, a następnie samodzielnie (ręcznie) uruchomić **MRTG** z linii poleceń. Wtedy mamy **pewność**, że zobaczymy co jest nie tak i szybko uda nam się poprawić dany błąd.



Szybkie podsumowanie statystyk



Mamy już ładne wykresy, polską wersję, usługę działającą bez naszego bezpośredniego udziału.
No tak, ale mamy wszystkie pliki w jednym folderze, co potęguje bałagan i nie świadczy o naszym profesjonalizmie.



rysunek 21. MRTG wszystko w jednym folderze.

Czas to poprawić.

Trzeba jeszcze trochę poprawić nasz plik konfiguracyjny.

Musimy zakomentować linię:

```
WorkDir: c:\mrtg\test
```

Komentarze wprowadza się poprzedzając linię znakiem (#)

```
#WorkDir: c:\mrtg\test
```

Linia powinna wyglądać tak.

Natomiast dopisać powinniśmy linijki:

```
IconDir:   c:\mrtg\www\icons  
HtmlDir:  c:\mrtg\www  
ImageDir: c:\mrtg\www\images  
LogDir:   c:\mrtg\www\logs
```

Zmienna **WorkDir** bezwzględnie nadpisuje powyższe zmienne, stąd konieczność jej wyłączenia z konfiguracji.

Przed uruchomieniem usługi należy stworzyć foldery:

```
c:\mrtg\www  
c:\mrtg\www\icons  
c:\mrtg\www\images  
c:\mrtg\www\logs
```

Po uruchomieniu otrzymamy logiczną strukturę danych:

W katalogu „*www*” znajdą się tylko pliki htm, w „*icons*” ikony biblioteki MRTG, w „*images*” wszystkie wykreowane wykresy, zaś w „*logs*” tekstowe pliki będące bazą danych dla operacji tworzenia wykresów.

Teraz jak na dłoni widać, że nasze statystyki to owoc przemyślanej pracy.

A i pozostała nam jeszcze jedna rzecz. Strona tytułowa.

Możemy sami stworzyć plik html z listą wszystkich wykresów, ale nie widzę takiej potrzeby jeśli istnieje do tego gotowa biblioteka w pakiecie MRTG.

Przy użyciu konsoli przemieszczamy się do katalogu:

```
c:\mrtg\bin
```

Wykonujemy następujące operacje:

```
perl indexmaker
    --output = c:\mrtg\www\index.html
    --columns = 1
    --prefix = ./
    --icondir = ./icons
test.conf
```

Generujemy plik „**index.html**” w katalogu, razem ze wszystkimi innymi podstronami.

Definiujemy ilość kolumn na 1, bowiem statystyki w takiej formie najładniej się prezentują przy dużych wykresach, przy standardowych 2 kolumny są w sam raz.

Następnie stwierdzamy, że życzymy sobie aby plik „**index.html**” został wygenerowany tam gdzie definiuje to zmienna **WorkDir** lub **HtmlDir**.

Dalej, z uwagi na nasze porządki, musimy zmienić katalog, w którym przechowywane są grafiki.

Na samym końcu podajemy, na podstawie którego pliku konfiguracyjnego stworzyć stronę z podsumowaniem.

W tej podstawowej konfiguracji strona zostanie stworzona sekcja po sekcji, chyba że innym parametrem wymusimy inną metodę.

Źródła danych i natchnienia



W stworzeniu opracowania pomogły:

- Manual MRTG by Tobias Oetiker
<http://oss.oetiker.ch/mrtg/doc/mrtg-reference.en.html>
- Manual MRTG – modułu indexmaker by Tobias Oetiker
<http://oss.oetiker.ch/mrtg/doc/indexmaker.en.html>
- Artykuł „SNMP MRTG for Windows” z serwisu [syslog.gr](http://www.syslog.gr)
<http://www.syslog.gr/articles-mainmenu-99/10-snmp-mrtg-for-windows.html>
- Artykuł „MRTG - czyli ładne statystyki pracy systemu” by Michał Płuciennikowski
<http://home.wecon.pl/~plucien/?Article=27>
- Posty na forach internetowych związanych z administrowaniem serwerami internetowymi
- Serwisowi Youtube.com za dostarczenie muzyki pomagającej napisać powyższe opracowanie.

Historia zmian:

<i>Data zmiany</i>	<i>Wersja</i>	<i>Zakres zmian</i>
16.05.2010	0.1	Pierwsza wersja dokumentu
19.05.2010	1.0	Zaopiniowana wersja dokumentu z naniesionymi poprawkami edycyjnymi.
19.05.2010	1.1	Poprawienie hiperłączy do paczek z danymi.