Monitoring Pracy Servera

Autorzy: Grzegorz Karnas, Wojciech Bednarz IVFDS

STRESZCZENIE

W projekcie zawarliśmy przykładowe rozwiązanie problemu, z którym spotyka się z pewnością wielu administratorów, a mianowicie:
Często bywa tak, że nasz server jest zwykłym PC-tem, udostępniającym routing pakietów, kilka jakichś demonów typu: proftpd, sshd, samba oraz to, co nam potrzebne. Zazwyczaj jest to jednostka pozbawiona monitora i jedyną drogą sprawdzenia poprawności działania naszego servera jest podłączenie do niego klawiatury, monitora i ręczne sprawdzenie tego, co się tam „dzieje", lub też logowanie się zdalne.
Ta praca rozwiązywałaby przynajmniej część informacyjna tego problemu, czyli przedstawiałby informacje dotyczące pracy servera.
Przykładowe informacje, które będziemy mogli uzyskać to:

- czas pracy servera od ostatniego reboota,
- listę najbardziej pamięcio-żernych aplikacji,
- procentowe zużycie mocy procesora,
- wykres czasowy zużycia mocy procesora.
- ilość wolnej pamięci operacyjnej, pamięci SWAP,
- ilość dostępnego miejsca na wszystkich podmontowanych partycjach,
- ilość ściągniętych i wysłanych danych oraz aktualna szybkość transferu danych wchodzących i wychodzących na terminalach,
- wiele innych tego typu danych (wszystko zależy już tylko i wyłącznie od wyobraźni administratora i jego umiejętności programowych oraz wiedzy na temat sposobu otrzymania interesujących go danych)

W tym momencie nasuwa się pytanie jak bylibyśmy informowani przez server o tych wszystkich rzeczach?
Do tego celu wykorzystamy wyświetlacz ciekłokrystaliczny (w tym przypadku 4 x 20 znaków), podłączony do servera przez port LPT.
Postaramy się przedstawić jak w prosty sposób możemy zaopatrzyć nasz Server w takie urządzenie, opiszemy sposób pisania własnych aplikacji czy skryptów pracujących na serwerze i zarządzających treścią komunikatów.

SPIS TREŚCI

1. PODŁĄCZENIE DO KOMPUTERA

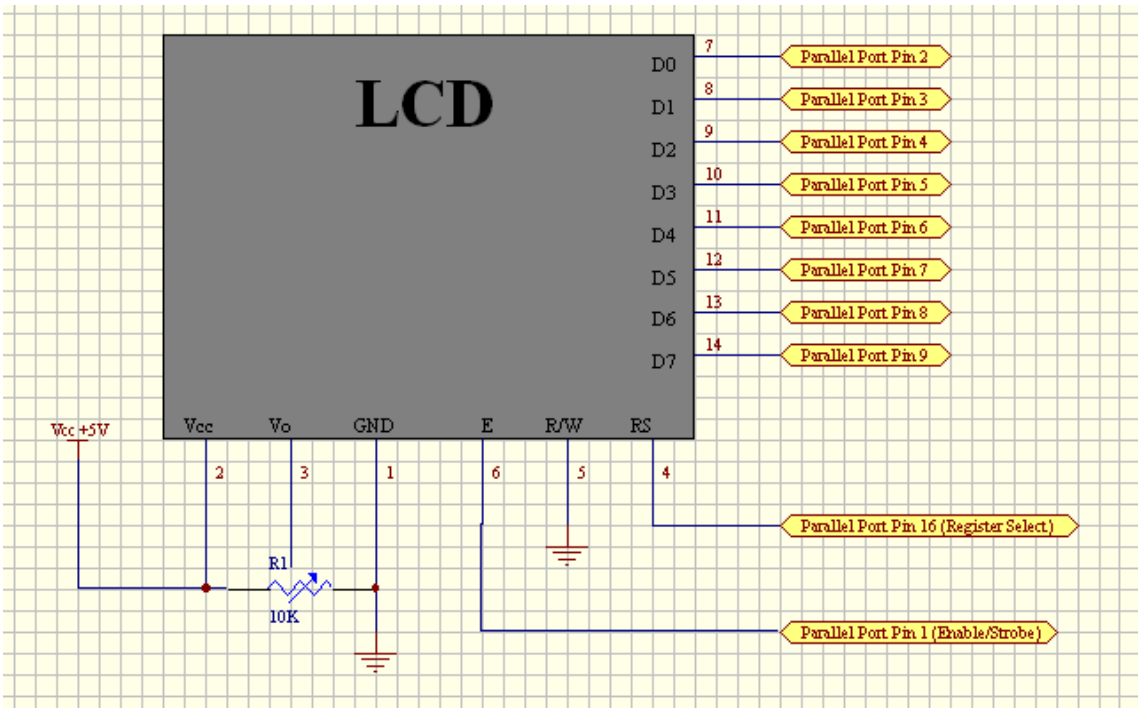
Alfanumeryczne wyświetlacze ciekłokrystaliczne LCD są urządzeniami coraz częściej wykorzystywanymi do monitoringu procesów lub jako forma interfejsu z użytkownikiem. Przykładami są choćby automaty do sprzedaży napojów czy aparaty telefoniczne firmy URMET. Posiadają one specjalizowany mikroprocesor, pamięć RAM, CGRAM, i ROM. Całością zarządza specjalizowany układ Hitachi HD 44780. Protokół jest bardzo przejrzysty i nie nastręcza wielu problemów podłączenie wyświetlacza do komputera PC.

1.1 Projekt LCDproc

Projekt ten zapoczątkowany w 1998 roku miał jako główne zadanie stworzenie w oparciu o koncepcję Open Source architektury klient-serwer umożliwiającej proste i uniezależnione od sprzętu wykorzystanie modułów LCD.

Główną stroną projektu jest http://lcdproc.omnnipotent.net**.** W tym opracowaniu przedstawię pakiet lcdproc-0.4.1. Nie jest to jego najnowsza wersja, ale na niej oprzemy cała pracę.

1.2 Schemat układu



Rys 1.2 Sposób podłączenia wyświetlacza do komputera PC za pośrednictwem portu LPT.

Jak widać sam schemat jest bardzo prosty do realizacji i nie powinien nastręczać kłopotów nawet niedoświadczonym hobbystom elektroniki.

1.3 Montaż

Po zapoznaniu się ze schematem układu i określeniu miejsca, w którym ma znajdować się wyświetlacz, pozostaje zaprojektować płytkę główną, która będzie łączyła nasz wyświetlacz z komputerem PC. Nie podajemy tu żadnego przykładu, gdyż istnieją setki możliwości zarówno umiejscowienia wyświetlacza jak i rozbudowy go o dodatkowe moduły typu joystick, czy odbiornik podczerwieni. Jedyną ważną rzeczą, o której musimy pamiętać w trakcie montażu jest to, że wyświetlacze LCD są bardzo delikatnymi i podatnymi na uszkodzenia urządzeniami, więc realizując nasz układ musimy zwrócić szczególną uwagę na prawidłowe podpięcie wszystkich elementów jak i na to, aby w trakcie lutowania nie przegrzać wyświetlacza.

1.4 Instalacja i konfiguracja oprogramowania

Jako, że projekt bazuje na systemie Unixowym, od jego administratora wymagana jest podstawowa wiedza dotycząca tych systemów i sposobu poruszania się po nich.

Proces instalacji możemy podzielić na kilka etapów:

Rozpakowanie archiwum:	`tar xfvz lcdproc-0.4.1.tar.gz`
Przejście do rozpakowanego katalogu:	`cd lcdproc-0.4.1`
Konfiguracja:	`./configure "--enable-drivers=hd44780”`
Kompilacja:	`make`
Uzyskanie praw root’a:	`su`
Instalacja:	`make install`

Uwagi: kompilacja wymaga bibliotekincurses.

Pakiet składa się z dwóch części serwera (LCDd) i klienta (lcdproc), wymagających skonfigurowania.

Konfiguracja serwera odbywa się przez podanie parametrów opisujących urządzenie obsługiwane przez sterownik:

```
LCDd –t 20x4 –d HD44780 „-c winamp – p 0x378”
```

-t	określa ilość kolumn x wierszy; wymagany jako pierwszy
-d	nazwa_sterownika
-c winamp	określa sposób podłączenia
-p 0x378	port na którym podłączony jest wyświetlacz

Serwer standardowo lokuje się na porcie 13666, możliwym do zmiany przy ./configure.

Klient lcdproc jest konfigurowany w ujęciu adresu serwera i portu (-s -p –domyślnie localhost i 13666) oraz screenów, które ma wyświetlać:

C – informacje o obciążeniu procesora
T – czas systemowy i uptime
M – informacje o zużyciu pamięci
X – Wykres obciążenia w osi czasu
D – informacje o dyskach
B – zużycie baterii
S – top 5-ciu najbardziej zużywających pamięć procesów

Pozostałe parametry są określane jako depreciated i w przyszłości mogą zostać pominięte.

Przykład:
lcdproc C T M X S &

W celu uruchamiania pakietu na starcie tworzymy odpowiedni skrypt w katalogu /etc/init.d

```
#! /bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
NAME=lcd
PARAMS='C T M X S'

case "$1" in
  start)
      echo -n "Starting LCDproc: LCDd "
      LCDd -t 20x4 -d HD44780 "-c winamp -p 0x378"
      echo "lcdproc."
      lcdproc $PARAMS &
      ;;
  stop)
      echo -n "Stopping LCDproc: lcdproc "
      killall -9 LCDd
      echo "LCDd."
      killall -9 lcdproc
      ;;
  restart|force-reload)
      echo "Restarting LCDproc"
      killall -9 lcdproc
      killall -9 LCDd
      sleep 1
      LCDd -t 20x4 -d HD44780 "-c winamp -p 0x378"
      lcdproc $PARAMS &
      ;;
  *)
      N=/etc/init.d/$NAME
      echo "Usage: $N {start|stop|restart|force-reload}" >&2
      exit 1
      ;;
esac

exit 0
```

Skrypt reaguje na parametry stop/start i restart.
Po utworzeniu odpowiednich dowiązań symbolicznych w etc/rc...:

```
ln –s /etc/init.d/lcd  /etc/rc[2].d/S50lcd
ln –s /etc/init.d/lcd  /etc/rc[3].d/S50lcd
ln –s /etc/init.d/lcd  /etc/rc[4].d/S50lcd
ln –s /etc/init.d/lcd  /etc/rc[5].d/S50lcd
ln –s /etc/init.d/lcd  /etc/rc[0].d/K50lcd
ln –s /etc/init.d/lcd  /etc/rc[6].d/K50lcd
```

wyświetlacz jest odpowiednio obsługiwany we wszystkich poziomach startu systemu.

2. PISZEMY KLIENTA

2.1 Protokół komunikacji klient-server

Serwer standardowo dostępny na porcie 13666 komunikuje się z klientami dwustronnie za pomocą prostego zestawu komend. W celach testowych można się „zatelnetować” na ten port.

Pierwszą komendą wymaganą do powiadomienia serwera o istnieniu nowego klienta jest hello, serwer odpowiada to opisując swoją wersję i charakteryzując urządzenie, które obsługuje:

connect LCDproc 0.4.1 protocol 0.3 lcd wid 20 hgt 4 cellwid 5 cellhgt 8

Pozostałe komendy:

`client_set [-name #id]` określa nazwę klienta

`screen_add #id` dodaje nowy screen

`screen_del #id` usuwa screen

```
screen_set #id [-priority integer] [-name "my_name"] [-duration integer]
         [-wid width] [-hgt height] [-heartbeat type]
```
inicjalizuje screen lub zmienia jego zawartość

Znak aktywności może być w trybie:

on, heart	ikona „serca”
normal, default	standardowa
off, none	wyłączone
slash	obracający się slash

Priorytet określany jest liczbą z przedziału 0-255; 0 oznacza najwyższy, zaś 255 najniższy. 64 jest domniemany.

```
widget_add #screen #id type [-in #id]
```
definiuje widget

Mogą być one typu: "string", "hbar", "vbar", "title", "icon" (nie zaimplementowany!), "scroller", "frame",

`widget_del #screen #id` usuwa widget

`widget_set #screen #id data` precyzuje zawartość, położenie i inne parametry widgetu

Parametry określające widgety:

string	x y tekst
hbar	x y długość w pixelach
vbar	x y długość w pixelach
icon	x y dane_binarne
title	tekst
scroller	left top right bottom direction speed text
	kierunek może być "h" lub "v".
frame	left top right bottom wid hgt dir speed

Komunikaty serwera:

`connect ....` określa wersję i rodzaj sprzętu

`bye` oznacza shutdown serwera.

`huh? [info]` sygnalizuje odebranie błędnej komendy

`listen [#screen [#widget [#widgets...]]]`
oznacza nasłuchiwanie danego klienta

`ignore [#screen [#widget [#widgets...]]]`
informuje, że klient powinien przestać wysyłać komendy

`key #id`
informacja o naciśnięciu klawisza.

Wśród dostępnych rozszerzeń pakiet istnieją interfejsy do obsługi joysticka i odbiornika podczerwieni (sterowanie pilotem).

2.2 Klient skryptowy

Screeny klientów są kolejkowane wedle zadanych priorytetów, możliwe jest więc użycie własnych rozszerzeń używając opisanego powyżej protokołu.

Napisane przez nas poniższe skrypty w Perlu obrazują łatwość wykorzystania tego mechanizmu:

Pierwszy skrypt pokazuje informacje o autorze i czas systemowy:

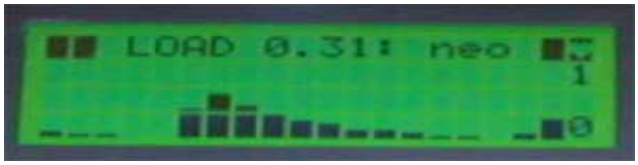
```
#!/usr/bin/perl -w
use IO::Socket;
use Fcntl;
# Connect to the server...
$remote = IO::Socket::INET->new(
            Proto     => "tcp",
            PeerAddr  => "localhost",
            PeerPort  => "13666",
      )
      || die "Cannot connect to LCDproc port\n";

# Make sure our messages get there right away
$remote->autoflush(1);

sleep 1;    # Give server plenty of time to notice us...

print $remote "hello\n";
my $lcdconnect = <$remote>;
print $lcdconnect;

# Turn off blocking mode...
fcntl($remote, F_SETFL, O_NONBLOCK);

# Set up some screen widgets...
print $remote "client_set name {NEO}\n";
print $remote "screen_add neo\n";
print $remote "screen_set neo name {neo} -heartbeat off\n";
print $remote "widget_add neo ktoco string\n";
print $remote "widget_add neo czas string\n";
print $remote "widget_set neo ktoco 3 1 'Grzegorz Karnas Wojciech Bed-
narz'\n";
print $remote "widget_add neo zczego scroller\n";
print $remote "widget_set neo zczego 1 2 20 2 h 2 ' ---== Projekt Monitoring
pracy servera ==--- '\n";

$d=":";
$wday=$yday=$isdst=0;
while(1)
{
select(undef,undef,undef,0.5);
$nowtime=time+3600; ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
gmtime($nowtime);
$year+=1900; $mon++; for ($sec,$min,$hour,$mday,$mon) {s/^(.)$/0$1/;}
print $remote "widget_set neo czas 7 4 '$hour$d$min$d$sec'\n";

if ($d eq ':') {$d=" ";} else {$d=":";}

}

close ($remote)            || die "close: $!";
exit;
```

Drugi skrypt podaje informacje o stanie komputerów w sieci lokalnej, jest to przeróbka skryptu iosock.pl:

```
#!/usr/bin/perl -w

use IO::Socket;
use Fcntl;

# Connect to the server...
$remote = IO::Socket::INET->new(
            Proto     => "tcp",
            PeerAddr  => "localhost",
            PeerPort  => "13666",
      )
      || die "Cannot connect to LCDproc port\n";

# Make sure our messages get there right away
$remote->autoflush(1);

`sleep 1`;  # Give server plenty of time to notice us...

print $remote "hello\n";
my $lcdconnect = <$remote>;
#print $lcdconnect;


# Turn off blocking mode...
fcntl($remote, F_SETFL, O_NONBLOCK);


# Set up some screen widgets...
print $remote "client_set name {Test Client (Perl)}\n";
print $remote "screen_add pings\n";
print $remote "screen_set pings name {Ping Status} -heartbeat off\n";
print $remote "widget_add pings onet string\n";
print $remote "widget_add pings twot string\n";
print $remote "widget_add pings one scroller\n";
print $remote "widget_add pings two scroller\n";
print $remote "widget_set pings onet 1 1 {Checking machines...}\n";


my ($machine, %down, %up, $i, $list);

while(1)
{
      #print "Main loop...\n";
      # Handle input...  (just spew it to the console)
      while(defined($line = <$remote>)) {
          if ( $line =~ /^success$/ ) {next;}
#         print $line;
      }

      undef %down;
      undef %up;
      foreach $machine       ("192.168.0.1",
                              "192.168.0.2",
                              "192.168.0.3",
                              )
      {
            `ping -c 1 $machine`;
            if($?) { $down{$machine} = 1; }
```

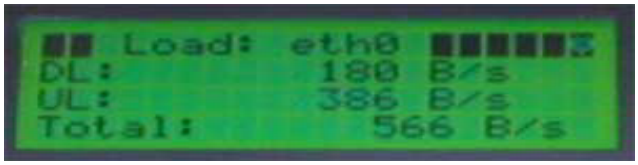
```
            else { $up{$machine} = 1; }
      }
      $i = 0;  $list = "";
      foreach $machine (keys %up)
      {
            $i++;
            $list .= "$machine ";
      }
      print $remote "widget_set pings onet 1 1 {Machines Up: ($i):}\n";
      print $remote "widget_set pings one 1 2 20 2 h 2 {$list}\n";

      $i = 0;  $list = "";
      foreach $machine (keys %down)
      {
            $i++;
            $list .= "$machine ";
      }
      print $remote "widget_set pings twot 1 3 {Machines Down: ($i):}\n";
      print $remote "widget_set pings two 1 4 20 4 h 2 {$list}\n";

}
close ($remote)            || die "close: $!";
exit;
```

3. PRZYKŁADOWE SCREENY

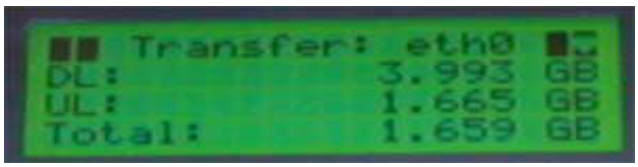


Rys 3.1 Wykres obciążenia procesora w funkcji czasu

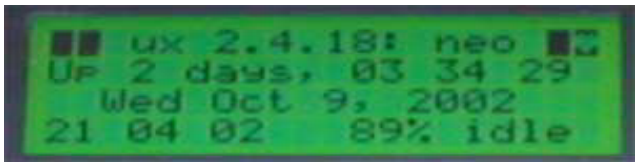
Rys 3.2 Procentowe obciążenie procesora



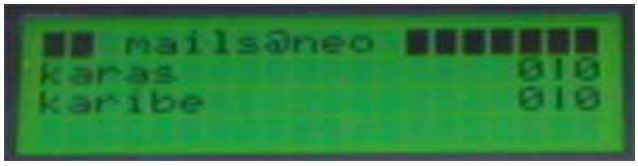
Rys 3.3 Aktualny transfer na terminalu eth0



Rys 3.4 Suma wysłanych i odebranych danych na terminalu eth0



Rys 3.5 Aktualna data i czas, oraz uptime servera



Rys 3.6 Ilość nowych i wszystkich maili administratorów

4. UWAGI I WNIOSKI

Jak widać samodzielny montaż jak i późniejsze użytkowanie wyświetlacza nie powinno sprawiać większego kłopotu. Po jakimś czasie pracy z wyświetlaczem, pisania własnych klientów można dojść do wprawy, a nawet pokusić się o kompletną przeróbkę układu na własne potrzeby użytkowe. Wszystkie przedstawione screeny pochodzą z komputera jednego z autorów tej publikacji.

Jako, że cały projekt opierał się o źródła internetowe, bardzo ciężko jest podać jakąś konkretną pozycję książkową traktującą o tym temacie. Mamy nadzieję, że zainteresowaliśmy Was nowymi możliwościami i spróbujecie nad tym popracować. Cały koszt takiego przedsięwzięcia zamyka się w 150 PLN. Najdroższym elementem jest oczywiście wyświetlacz, którego cena waha się w granicy około 100-120 PLN (w zależności od wielkości i od tego czy ma wbudowane podświetlanie). Mimo to życzymy udanych eksperymentów.

5. LITERATURA

[1] http://lcdproc.omnipotent.net

[2] http://freshmeat.net