

Portable Perl Daemons

Timothy Procter, procter@pythian.com

May 21, 2009

DAEMON: *Disk And Execution MONitor*

- **Situation**

- Application already widely distributed
- Client scheduled periodically (crontab)

- **Goals**

- Continuous background parent process
- Support for most UNIX and Windows systems
- Require NO manual intervention per client

- **Limitations**

- OS Implementation Differences
- Distribution of Modules

Unix Daemon Method

```
use POSIX qw(setsid);

# daemonize the program
&daemonize;

# Main program LOOP
while(1) {
    do_stuff();

    # Pace yourself!
    sleep(N);
}

sub daemonize {
    chdir '/' or die "Can't chdir to /: $!";
    open STDIN, '/dev/null' or die "Can't read /dev/null: $!";
    open STDOUT, '>>/dev/null' or die "Can't write to /dev/null: $!";
    open STDERR, '>>/dev/null' or die "Can't write to /dev/null: $!";
    defined(my $pid = fork) or die "Can't fork: $!";
    exit if $pid;
    setsid or die "Can't start a new session: $!";
    umask 0;
}
```

Windows Alternative

```
use Win32::Process;  
use Win32;  
  
Win32::Process::Create(  
    my $ProcessObj,  
    "c:\\perl\\bin\\perl.exe",  
    "perl 'script.pl' '-configfile bddvlp.cfg'",  
    0,  
    DETACHED_PROCESS,  
    "C:\\path\\to\\perl\\"  
) or die Win32::FormatMessage( Win32::GetLastError() );
```

They didn't give me a fork so I have to eat with a spawn.

Obstacles to Portability

- fork
- sleep
- /dev/null
- setsid
- Win32::Process

Steps

1. Create a child process
2. Ensure single process
3. Detach from parent
4. Main Loop
5. Child Processes

Create a Child Process

```
if (!@ARGV or $ARGV[-1] ne '-dmon') {
    #like a wrapper script - detach from terminal, replace self
    print "Detaching from parent process...\n";
    if ($^O ne 'MSWin32') {
        my $pid = fork();
        if (!$pid) {
            exec qq{perl "$0" @ARGV -dmon};
        }
    }
}
else {
    require Win32::Process;
    my $proc;
    Win32::Process::Create(
        $proc,                # Process object
        qq{$^X},             # Command
        "perl $0 @ARGV -dmon", # Parameters
        0,                   # Inherit handles?
        Win32::Process->DETACHED_PROCESS, # Detach from console
        '.');                # Current directory
}
exit;
}
```

Ensure Singular Process

```
#control by lockfile
my $lockfile = '.hlock';
if (-r $lockfile) {
    if (open (my $hlock, '<', $lockfile)) {
        if (defined(my $lock_pid = <$hlock>)) {
            chomp($lock_pid);
            if (kill (0, $lock_pid)) {
                print
                "Process $lock_pid is already running.\n";
                close $hlock;
                exit 0;
            }
        }
    }
}
open (my $hlock, '>', $lockfile)
    or die "Cannot open lockfile '$lockfile' $!";
print $hlock $$;
close $hlock;
```

Detach From Parent

```
sub detach {
  my $nul = $^O eq 'MSWin32'
    ? 'NUL' : '/dev/null';
  open STDIN, '<', $nul or die "Can't read NUL: $!";
  open STDOUT, '>', '&STDIN' or die "Can't dup STDIN: $!";
  open STDERR, '>', '&STDOUT' or die "Can't dup STDOUT: $!";
  if ($^O ne 'MSWin32') {
    umask 0;
    setsid;
  }
  #chdir '/' or die "Can't chdir to /: $!";
}
```

Main Loop

```
#sub main {  
  
while (1) {  
    foreach my $channel (values %channel_of) {  
        #get_response(retry_iteratations)  
        if (my $msg = $channel->get_response(0)) {  
            process_msg($channel, @$msg);  
        }  
    }  
  
    #Sleep  
    select(undef, undef, undef, $MAIN_SLEEP_INT);  
  
    #periodically, check for ADM connections  
    if (my $new_cntn = $h_server->nb_accept()) {  
        $channel_of{$ADM}{'channel'} = $new_cntn;  
        $log->log(0, $INFO_MSG,  
            'Accepted ADM Connection.');    }  
}  
}
```

Child Processes

```
foreach my $child (@supporting_processes) {  
    # Fork agent process  
    my $pid;  
    if ($^O eq 'MSWin32') {  
        Win32::Process::Create(  
            # Similar to parent process  
        );  
        $pid = $proc->GetProcessID();  
    }  
    else {  
        $pid = fork();  
        if (!$pid) {  
            exec "perl agent.pl localhost $lport $agent";  
        }  
    }  
}
```

- Resulting Process Hierarchy

```
\_ perl handler_ctl.pl  
\_ perl handler.pl -dmon  
    \_ perl agent.pl  
        \_ perl agent.pl
```

SIGHUP Signal

- Capture and handle Interrupts

```
$SIG{HUP} = sub {  
    #restart/reload  
    exec qq{perl "$0" @ARGV};  
};  
  
#$SIG{INT} ...
```

Inter-process Communication

- Signals
- Named Pipes
- `open()`: STD*
- Sockets
- File-sharing
- Variable-sharing (threads)

Long-term Behaviour

- (I don't know)
- Memory can be a problem
 - Minimize libraries used in main process
 - Monitor processes

Resources

- <http://search.cpan.org/~nwclark/perl-5.8.9/pod/perlipc.pod>
- <http://www.pythian.com/news/wp-content/uploads/daemons-demo.zip>
 - Channel
 - Log

Questions?