Instytut Teleinformatyki

Wydział Inżynierii Elektrycznej i Komputerowej
Politechnika Krakowska

programowanie usług sieciowych

„Demonizacja i logowanie”

laboratorium: 05

Kraków, 2014

Spis treści

1. Wiadomości wstępne

Proces - to jedno z najbardziej podstawowych pojęć w informatyce. Z definicji jest to po prostu egzemplarz wykonywanego programu. Należy odróżnić jednak proces od wątku - każdy proces posiada własną przestrzeń adresową, natomiast wątki posiadają wspólną sekcję danych.

Każdemu procesowi przydzielone zostają zasoby, takie jak:

- procesor,
- pamięć,
- dostęp do urządzeń wejścia-wyjścia,
- pliki.

Demonem(ang. *daemon*) nazywamy proces uruchomiony w tle, który nie podlega sterowaniu z żadnego terminala. W systemie Unix, wykonuje w tle wiele procesów będących demonami. Zajmują się one rozmaitymi sprawami związanymi z zarządzaniem systemem.

Procesy-demony można uruchamiać w różny sposób:

- odpalanie demonów za pomocą skryptów wykonywalnych, umieszczonych najczęściej w katalogu `/etc/rc` lub `/etc/init.d/`,
- uruchamianie serwerów sieciowych za pomocą serwera nadrzędnego `xinetd`,
- za pomocą demona `cron`, który uruchamia programy co określony czas, plik konfiguracyjny umieszczony w `/var/spool/cron/crontabs/root`,
- poprzez polecenie `at`, które uruchamia program w pewnym określonym terminie, przybliżona składnia `at [-opcje] [-f file] [-opcje] TIME`,
- uruchomienie demonów poprzez konsole terminala.

1.1. Tematyka laboratorium

Tematyką naszego laboratorium będzie:

- demonizacja procesów,
- zasada działania demona `syslogd`,
- rejestrowanie komunikatów przez demony, za pomocą funkcji `syslog`,
- zasada działania i konfiguracja demona `xinetd`.

1.2. Zagadnienia do przygotowania

Przed przystąpieniem do laboratorium należy się zapoznać z zagadnieniem demonizacji. Szczególną uwagę należy zwrócić na **demony**:

- *syslogd*; [Stevens, Rozdział 12]
- *inetd*; [Stevens, Rozdział 12]

oraz przypomnieć sobie podstawowe funkcje w języku C, do działania na procesach i demonach.

Literatura:

[1] W.R. Stevens, *„Programowanie Usług Sieciowych”*, *„API: gniazda i XTI”*.
[2] Dokumentacja NetBSD (http://doc.netbsd.pl/), rozdział 15.
[3] http://linio.terramail.pl/xinetd.html - opis pliku konfiguracyjnego.

1.3. Cel laboratorium

Celem laboratorium jest zapoznanie się z zastosowaniem demonów w systemie operacyjnym Unix. W szczególności ze sposobami rejestrowania komunikatów wysyłanych przez demony, czyli działaniem demona `syslogd` oraz działaniem serwera `xinetd`. Podczas realizacji tego laboratorium zapoznasz się z:

- ogólną zasadą demonizacji dowolnego procesu,
- zasadą rejestrowania logów w systemie Unix,
- z możliwościami konfiguracji demona `syslogd` oraz zasadami jego działania,
- z możliwościami konfiguracji demona `xinetd` oraz zasadami jego działania.

2. Przebieg laboratorium

Ta część instrukcji zawiera zadania do praktycznej realizacji, które demonstrują zastosowanie technik z omawianego zagadnienia.

2.1. Przygotowanie laboratorium

Należy zalogować się na stanowisku jako root a następnie przekopiować zawartość katalogu

`Home/shared/pus/pus_05_demon`

do katalogu:

`~/pus/pus_05_demon`

`~/pus/pus_05_demon`

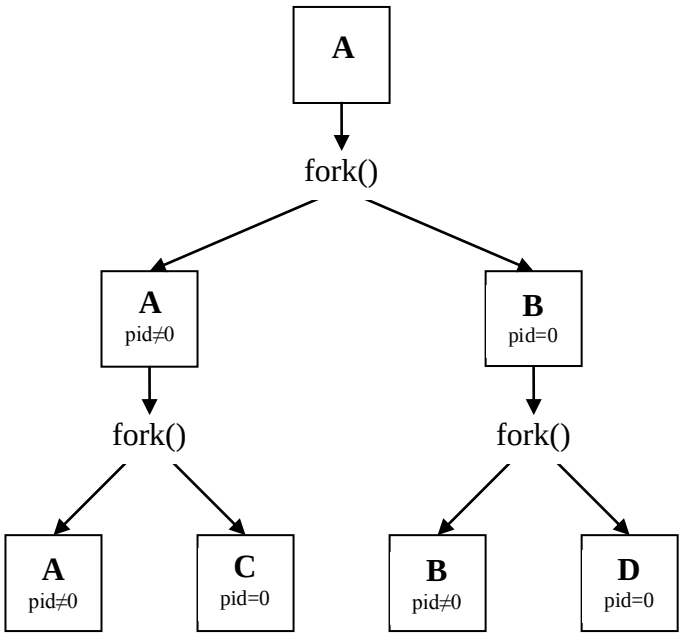
2.2. Zadanie 1. Funkcja fork()

Działanie funkcji prezentuje program fork_example.c

```
int main(int argc, char *argv[])
{
      int pid;
      char a='A';
      printf("%c wykonuje proces fork\n",a);
      pid = fork();
      if (pid==0) a=a+1;
        printf("%c ma pid: %d\n", a,pid);
        sleep(1);
        printf("%c wykonuje drugi proces fork\n",a);
        pid = fork();
      if (pid==0) a=a+2;
      printf("%c teraz ma pid: %d\n",a, pid);
        sleep(1);
}
```

Proces oznaczony jako A wywołuje funkcję fork (pid procesu A przyjmuje wartość identyfikatora procesu B) tworząc w ten sposób proces potomny B (o wartości

pid=0). Kolejne wywołanie funkcji fork() jest już wykonywane przez oba procesy. Proces A tworzy kolejnego potomka, oznaczonego jako C (pid procesu A zmienia się na wartość równą identyfikatora procesu C, pid procesu C jest zerowy, ponieważ jest to potomek). Proces B również tworzy potomka oznaczonego jako D stając się w ten sposób rodzicem (jego pid z wartosci 0 zmienia się na wartość równą identyfikatorowi procesu D). Proces ten jest zilustrowany na poniższym rysunku.



Rys. 6. Działanie funkcji fork()

1. Skompiluj plik źródłowy fork_example.c:
   ```
   $ gcc fork_example.c -o fork_example
   ```
2. Uruchom skompilowany program:
   ```
   $ ./fork_example
   ```
   na ekranie powinien pojawić się wynik zbliżony do takiego (nie zerowe wartości pid będą różne od poniższych):
   ```
   A wykonuje proces fork
   B ma pid: 0
   A ma pid: 31647
   A wykonuje drugi proces fork
   B wykonuje drugi proces fork
   C teraz ma pid: 0
   A teraz ma pid: 31648
   D teraz ma pid: 0
   B teraz ma pid: 31649
   ```
3. Przeanalizuj działanie programu. Sprawdź co by się stało gdyby po warunkach
   ```
   if (pid==0)...;
   ```


dodano linię

```
else exit(0);
```

Ile powstało procesów? Dlaczego tylko jeden przetrwał do końca działania programu?

2.3. Zadanie 2. Prosty serwer WWW

Aby uruchomić serwer WWW pracujący pod kontrolą demona `xinetd`, należy napisać dopisać odpowiednią dyrektywę do pliku konfiguracyjnego `/etc/xinetd.conf`.

Nasz serwer WWW jest skryptem shell'a wypisującym na standardowe wyjście odpowiedź HTTP 200 OK + zawartość strony.

W katalogu `/tmp` należy utworzyć plik serwerek.sh o następującej treści:

```
#!/bin/bash
read line
cat << EOF
HTTP/1.0 200 OK
Content-type: text/html
<html>
    <head>
        <title>Pierwszy serwer pod xinetd</title>
    </head>
    <body>
        <h1> Witamy na najprostszym serwerze WWW </h1>
    </body>
</html>
EOF
```

Następnie dopisujemy na końcu pliku `/etc/xinetd.conf`

```
service pus_service
{
    id          = serwerek
    type        = UNLISTED
    port        = 9000
    disable     = no
    socket_type = stream
    user        = root
    wait        = no
    server      = /tmp/serwerek.sh
}
```


Argument	Opis
`id`	unikalny identyfikator serwisu
`type`	wykorzystywane aby wskazać demonowi xinetd że jest to jeden z serwisów nie wymienionych w pliku /etc/services
`port`	port na którym będzie nasłuchiwał nasz serwer
`disable`	czy serwis jest dostępny
`user`	z jakim UID ma proces serwera
`wait`	czy należy czekać z uruchomieniem kolejnej instancji serwera przed zakończeniem poprzedniej
`server`	ścieżka do programu serwera obsługującego tę usługę

Tabela 1. Użyte atrybuty

Następnie restartujemy `xinet`:

```
rc.xinetd restart
```

W zalezności od wersji linuxa, plik znajdziemy pod nazwą:

- `/etc/rc.d/rc.xinetd` - Slackware,
- `/etc/init.d/xinetd` - Fedora.

Argument	Opis
`start`	uruchomienie demona xinetd
`stop`	zatrzymanie demona xinetd
`restart`	zatrzymanie i ponowne uruchomienie demona xinetd

Tabela 2. Skrypt posiada następujące argumenty:

Komunikaty od demona `xinetd` będą zapisywane do: pliku: `/var/log/servicelog`

W pliku `xinetd.conf` musi istnieć wpis w sekcji defaults:

```
log_type = FILE /var/log/servicelog
```

`log_type` określa sposób logowania zdarzeń związanych z dostępem do poszczególnych usług.

W pliku `/var/log/messages` zapisywane komunikaty diagnostyczne

Sprawdzamy w logach czy wszystko w porządku:
`$> tail var/log/messages`

```
Mar  8 14:43:10 serwer xinetd[2215]: xinetd Version 2.3.14 started
with libwrap options compiled in.
Mar  8 14:43:10 serwer xinetd[2215]: Started working: 2 available
services
```

W przypadku nieprawidłowej konfiguracji, otrzymamy informacje pozwalające szukać przyczyny błędu.
Jeżeli wszystko jest w porządku, przystępujemy do sprawdzenia działania serwera. Uruchamiamy ulubioną przeglądarkę WWW i wpisujemy adres naszego serwera:
`http://<IP>:9000`

2.4. Zadanie 3. Podstawy działania demona syslogd

Zadanie to pozwala na zapoznanie się ze strukturą pliku konfiguracyjnego `/etc/syslog.conf` oraz strukturą plików z logami.

1. Przejdź do katalogu ect:
   `$ cd /ect`
2. Otwórz plik konfiguracyjny:
   `$ more ./syslog.conf`
3. Przeglądnij jego zawartość, zapoznaj się ze strukturą pliku.
4. Znajdź informacje o tym, do jakiego pliku lub do jakich plików są wysyłane komunikaty dotyczące serwera poczty.
5. Przejdź do katalogu `/var/log`:
   `$ cd /var/log`
6. Przeglądnij wybrane pliki z logami. Zapoznaj się z ich strukturą.

2.5. Zadanie 4. Prosty program uruchamiany jako demon

Zadanie będzie polegało na zaznajomieniu się z kodem programu `demon.c`. który po uruchomieniu staje się programem demonem.

Program `demon.c` wykonuje wszystkie czynności potrzebne do tego, by stal się demonem. Wywołuje funkcje `fork`, aby proces potomny działał w tle, wywołuje funk-

cje `setsid`, aby utworzyć nowa sesje i zostać przywódcą tej sesji. Następnie ponownie wywołuje funkcje `fork`, aby uniknąć otrzymania nowego terminala sterującego, zmienia katalog roboczy oraz maskę trybu dostępu do tworzonych plików, wreszcie zamknąć niepotrzebne pliki.(patrz kod programu!)

1. Przejdź do katalogu `~/pus/pus_05_demon`:
   ```
   $ cd ~/pus/pus_05_demon
   ```
2. Skompiluj program `demon.c` poleceniem
   ```
   $ cc demon.c –o demon
   ```
3. Uruchom program:
   ```
   $ ./demon
   ```
4. Sprawdź czy nasz program rzeczywiście jest demonem. Możesz to sprawdzić poleceniem:
   ```
   $ ps –aux
   ```
   sprawdź czy na liście pojawił się nasz demon
5. W programie głównym wywołujemy funkcje
   ```
   syslog(LOG_INFO|LOG_LOCAL2,"%s działa jako demon",argv[0]);
   ```
   która rejestruje komunikat. Aby sprawdzić gdzie nasz komunikat został zapisany musimy zobaczyć w pliku /`etc/syslog.config` jakie SA reguły dla zapisywania naszego typu komunikatów, a następnie sprawdzić w odpowiednim pliku czy nasza wiadomość została dodana.

6. Po wykonaniu zadania należy zabić nasz proces demon poleceniem:
   ```
   $ kill <id_procesu>
   ```

2.6. Zadanie 5. Deamon TCP

Zadanie polega na modyfikacji załączonego kodu programu

`~/pus/pus_05_demon/server_tcp.c`

tak aby działał jako deamon.

W celu sprawdzenia poprawności wykonania zadania uruchom napisany daemon serwera TCP oraz program klienta TCP.

```
$ cd ~/pus/pus_05_demon/
$ gcc client_tcp.c -o client_tcp
$ ./client_tcp
```


2.7. Zadanie 6. Logi

Zadanie polegać będzie na modyfikacji demona z poprzedniego zadania, tak aby zapisywał logi o każdym nowym połączeniu klietna. Potrzebna do tego będzie funkcja syslog, opisana w opisie laboratorium.

3. Opracowanie i sprawozdanie

Realizacja laboratorium pt. „Demonizacja i logowanie” polega na wykonaniu wszystkich zadań programistycznych podanych w drugiej części tej instrukcji. Wynikiem wykonania powinno być sprawozdanie w formie wydruku papierowego dostarczonego na kolejne zajęcia licząc od daty laboratorium, kiedy zadania zostały zadane. Sprawozdanie powinno zawierać:

- opis metodyki realizacji zadań (system operacyjny, język programowania, biblioteki, itp.),
- algorytmy wykorzystane w zadaniach (zwłaszcza, jeśli zastosowane zostały rozwiązania nietypowe),
- opisy napisanych programów wraz z opcjami,
- trudniejsze kawałki kodu, opisane tak, jak w niniejszej instrukcji,
- uwagi oceniające ćwiczenie: trudne/łatwe, nie/realizowalne podczas zajęć, nie/wymagające wcześniejszej znajomości zagadnień (wymienić jakich),
- wskazówki dotyczące ewentualnej poprawy instrukcji celem lepszego zrozumienia sensu oraz treści zadań.

