Uzyskiwanie od systemu informacji o procesach
Zadanie

Jakub Szczyrk
235477

Poniedzałek 13:15-14:00

2 grudnia 2018

Spis treści

1 Zadanie 1

1.1 A. wyświetlenie komunikatu i zakończenie pracy programu

```
#include <time.h>
#include <signal.h>
#include <unistd.h>
#include <ucontext.h>
#include <stdlib.h>

void sighand(int sig, siginfo_t *sip, ucontext_t * uap)
{
printf("Dostalem signal numer %d\n",sig);
if(sip->si_code <= 0)
        printf("Od procesu %d\n", sip->si_pid);
printf("Kod sygalu %d\n",sip->si_code);
exit(0);
}
int main()
{
struct sigaction act;
act.sa_handler = sighand;
sigemptyset(&act.sa_mask);
act.sa_flags = SA_SIGINFO;
sigaction(SIGALRM, &act, 0);
sigaction(SIGTERM, &act, 0);
sigaction(SIGUSR1, &act, 0);
sigaction(SIGUSR2, &act, 0);

struct timespec ts;
ts.tv_sec = (time_t)0;
ts.tv_nsec = 100000001;

        for(int i=1; i; i++)
        {
                printf("Hello World!\n");
                nanosleep(&ts, NULL);
        }
return 0;}
```

1.2 B. wyświetlenie komunikatu i powrót do wykonywania programu

```
#include <time.h>
#include <signal.h>
#include <unistd.h>
#include <ucontext.h>

void sighand(int sig, siginfo_t *sip, ucontext_t * uap)
{
printf("Dostalem signal numer %d\n",sig);
if(sip->si_code <= 0)
        printf("Od procesu %d\n", sip->si_pid);
printf("Kod sygalu %d\n",sip->si_code);
}
int main()
{
struct sigaction act;

act.sa_handler = sighand;
sigemptyset(&act.sa_mask);
act.sa_flags = SA_SIGINFO;
sigaction(SIGALRM, &act, 0);
sigaction(SIGTERM, &act, 0);
sigaction(SIGUSR1, &act, 0);
sigaction(SIGUSR2, &act, 0);

struct timespec ts;
ts.tv_sec = (time_t)0;
ts.tv_nsec = 100000001;

        for(int i=1; i; i++)
        {
                printf("Hello World!\n");
                nanosleep(&ts, NULL);
        }
return 0;
}
```

1.3 C. wstrzymywanie odebrania sygnału (np. na 1000 iteracji w pętli), i następnie wznowienie odebrania sygnału

```
#include <time.h>
#include <signal.h>
#include <unistd.h>
#include <ucontext.h>
#include <stdlib.h>
#include <stdio.h>


void sighand(int sig, siginfo_t *sip, ucontext_t * uap)
{
printf("Dostalem signal numer %d\n",sig);
if(sip->si_code <= 0)
        printf("Od procesu %d\n", sip->si_pid);
printf("Kod sygalu %d\n",sip->si_code);
signal(sig,SIG_IGN);
for(int i =0; i<1000;i++)
{
}
signal(sig,SIG_DFL);
kill(fork(),sig);
}
int main()
{
struct sigaction act;

act.sa_handler = sighand;
sigemptyset(&act.sa_mask);
act.sa_flags = SA_SIGINFO;

sigaction(SIGALRM, &act, 0);
sigaction(SIGTERM, &act, 0);
sigaction(SIGUSR1, &act, 0);
sigaction(SIGUSR2, &act, 0);

struct timespec ts;
ts.tv_sec = (time_t)0;
ts.tv_nsec = 100000001;
        for(int i=1; i; i++)
        {
                printf("Hello World!\n");
                nanosleep(&ts, NULL);
        }
return 0;
}
```

1.4 D. całkowite ignorowanie sygnału

```
#include <time.h>
#include <signal.h>
#include <unistd.h>
#include <ucontext.h>
#include <stdlib.h>
#include <stdio.h>

void sighand(int sig, siginfo_t *sip, ucontext_t * uap)
{
printf("Dostalem signal numer %d\n",sig);
if(sip->si_code <= 0)
        printf("Od procesu %d\n", sip->si_pid);
printf("Kod sygalu %d\n",sip->si_code);
pause();
for(int i =0; i<1000;i++)
{}
}
int main()
{
struct sigaction act;

act.sa_handler = sighand;
sigemptyset(&act.sa_mask);
act.sa_flags = SA_SIGINFO;
sigaction(SIGALRM, &act, 0);
sigaction(SIGTERM, &act, 0);
sigaction(SIGUSR1, &act, 0);
sigaction(SIGUSR2, &act, 0);
signal(SIGALRM, SIG_IGN);
signal(SIGTERM, SIG_IGN);
signal(SIGUSR1, SIG_IGN);
signal(SIGUSR2, SIG_IGN);

struct timespec ts;
ts.tv_sec = (time_t)0;
ts.tv_nsec = 100000001;
        for(int i=1; i; i++)
        {
                printf("Hello World!\n");
                nanosleep(&ts, NULL);
        }
return 0;
}
```

2 Narzędzie proc

Proc jest pseudosystemem plików, który udostępnia interfejs do struktur danych jądra. Jest montowany w katalogu /proc.
Za pomocą niego możemy podejrzeć działajace procesy.

Jeżeli wyołamy takie polecenie z trzema ptokami poleceń

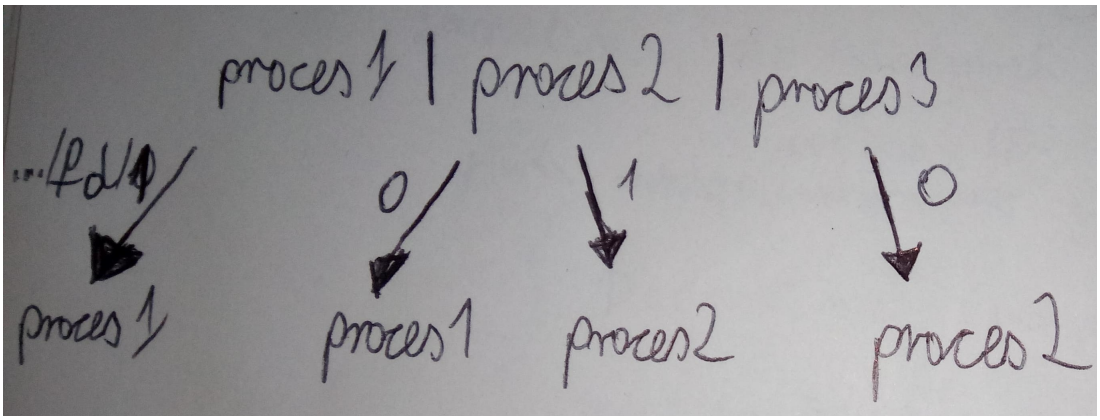
```
./zad2.sh | ./zad2b.sh | ./zad2c.sh
```

to wykonującym się procesem będzie "./zad2c.sh".

/proc/[pid]/fd/
Jest to podkatalog zawierające pliki:

- 0 jest standardowym wejściem,
- 1 jest standardowym wyjściem,
- 2 jest standardową diagnostyką
- 255 jest zawartością kodu wykonywanego procesu (zmieniając w nim treść, zmieniamy w orginale)

Co ciekawe dla ptoków jeżeli wejdziemy sobie w katalog fd/ to dla procesu "./zad2c.sh" w pliku 0 wyświetli nam sie program z "./zad2b.sh".
Ogólnie wyląda to tak jak na schemacie:



Kolejność występowania poleceń w potoku można sprawdzić właśnie powyższym sposobem.