Kamil Borkowski WCY22IY1S1 83374

zad 2 sygnaly2

Kod źródłowy:

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <time.h>

#include <signal.h>

int obudzony = 0;//zmiena moniturorujaca obudzenie sie pierwszego dziecka

void handle\_signal(int signum)

{

obudzony=1;//po otrzymaniu sygnalu od pierwszego obudzonego procesu potomnego czas zabic pozostale

}

int main(int argc, char \*argv[])

{

int n = atoi(argv[1]);//int n jest wprowadzane przy wywolaniu programu

srand(time(0));

int x[n];// czas sleep forkow

int i;// zmienna dla petli for

pid\_t child\_pids[n];

for (i = 0; i < n; i++)

{

usleep(1);

x[i] = (rand() % 8) + 2; // losowanie argumentu sleep od 2 do 9s

pid\_t pid = fork();

if (pid == 0)

{

printf("Proces %d (%d); wartość wylosowana: %d\n", i + 1, getpid(), x[i]);

sleep(x[i]);//uspienie dziecka na wylosowana ilosc sekund

printf("Obudził się proces %d (%d)\n", i + 1, getpid());

//wysyłanie sygnału SIGTERM do procesu pierwotnego

kill(getppid(), SIGTERM);

exit(0);

}

else

{

child\_pids[i] = pid;//dodanie pidu forka do tablicy forkow do zabicia

}

}

//wysalnie sygnalu po przebudzeniu dziecka

signal(SIGTERM, handle\_signal);

int pom=1;

while(pom)//czekanie na sygnal od pierwszego obudzonego procesu potomnego

{

if(obudzony)

{

for (int j = 0; j < n; j++)

{

kill(child\_pids[j], SIGTERM);

int status;

pid\_t terminated\_pid = waitpid(child\_pids[j], &status, 0);

if (!WIFEXITED(status))//sprawdzenie czy proces skonczyl sie sam czy zostal zabity

{

printf("Usunięto proces %d (%d)\n", j + 1, terminated\_pid);

}

}

pom=0;

}

}

return 0;

}

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie