|  |  |  |
| --- | --- | --- |
| |  | | --- | | **INSTYTUT TELEINFORMATYKI I AUTOMATYKI**  **Wydział Cybernetyki WAT** | | |
| |  | | --- | | Przedmiot: SYSTEMY OPERACYJNE  SPRAWOZDANIE Z ĆWICZENIA LABORATORYJNEGO NR 2 | | |
| **Temat**: Sygnały | |
| **Wykonał:**  xxxxx | **Data wykonania ćwiczenia:**  xxxxx  **Prowadzący ćwiczenie:**  xxxxx |

1. **Treść zadania laboratoryjnego**

Obraz zawierający tekst

Opis wygenerowany automatycznie

1. **Opis rozwiązania, komentarze, wnioski**

Program:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

int ppid0,pid01,pid02;

int b=1;

void SIGhandlerGET(int s){

    printf("\nI received input signal: %d\n",s);

    signal(s, SIG\_IGN);

}

void SIGINThandlerSEND(int s){

    printf("\nI've send SIGTERM signal to parent process: %d\n",ppid0);

    kill(ppid0, SIGTERM);

}

void SIGTERMhandlerRECEIVE(int s){

    printf("\nParent process received SIGTERM signal\n", ppid0);

    kill(pid01, SIGKILL);

    kill(pid02, SIGKILL);

    b=0;

}

void SIGhandlerEND(int s){

    signal(s, SIG\_IGN);

    b=0;

}

int main() {

    setbuf(stdout, NULL);

    ppid0=getpid();

    printf("PPID0: %d \n",ppid0);

    sigset\_t m;

    sigemptyset(&m);

    sigaddset(&m, SIGILL);

    sigprocmask(SIG\_SETMASK, &m, NULL);

    int i;

    if((pid01=fork())==0){

        printf("PID01: %d \n",getpid());

        sigemptyset(&m);

        sigprocmask(SIG\_SETMASK, &m, NULL);

        while(b){

            for(i=1; i<=32; i++)

                signal(i\*2, SIGhandlerGET);

            signal(SIGKILL, SIGhandlerEND);

        }

    }

    else

        if((pid02=fork())==0){

            printf("PID02: %d \n",getpid());

            sigemptyset(&m);

            sigprocmask(SIG\_SETMASK, &m, NULL);

            while(b){

                printf(".");

                sleep(1);

                signal(SIGINT, SIGINThandlerSEND);

                signal(SIGKILL, SIGhandlerEND);

            }

        }

    while(b)

        signal(SIGTERM, SIGTERMhandlerRECEIVE);

    return 0;

}

Konstrukcja funkcji sigprockmask:

Sigprockmask(int how, cinst sigset\_t \*set, sigset\_t \*oldset)

* how – określa tryb pracy (dodaj, usuń lub zmień maskę)
* \*set – sygnał lub sygnały, na których będą operacje
* \*oldset – jeśli NULL – nie robi nic, w przeciwnym razie zapisuje dotychczasową maskę w zadany adres

Inne używane funkcje:

* sigemptyset(&m) – czyści maskę m
* sigaddset(&m, s) – dodaje sygnał s do maski m

Działanie programu:

1. Powołane są proces macierzysty oraz 2 procesy potomne.
2. Jeden z procesy potomnych jako swoje działanie wysyła „.” co 1 sekundę.
3. Poprzez polecenie „kill -2 [nr dla pid01]” proces „pid01” otrzymuje sygnał „SIGINT”
4. Poprzez polecenie „kill -2 [nr dla pid02]” proces „pid02” otrzymuje sygnał „SIGINT”, w wyniku czego proces macierzysty otrzymuje sygnał SIGTERM.
5. Po informacji o otrzymaniu sygnału przez proces macierzysty następuje zamknięcie procesów potomnych.
6. **Wyniki uruchomienia programu**

Obraz zawierający tekst, zrzut ekranu, sprzęt elektroniczny, komputer

Opis wygenerowany automatycznie

