Programowanie współbierzne

Laboratoria 2

Piotr Stachnio 241268

18.03.2020

grupa zajęciowa nr. 2

Celem laboratoriów była nauka podstaw tworzenia procesów potomnych w systemach tupy unix za pomocą funkcji fork, execl oraz system.

Zadanie nr.1

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


void main(int argc, char **argv){

    int pid, status;
    int args[argc - 1];

    for(int i = 1; i < argc; i++){
        args[i - 1] = atoi(argv[i]);
    }

    for(int i = 0; i < argc - 2; i++){//jak wiele procesow chce utworzyc
        if((pid = fork()) == 0){

            for(int y = 0; y < args[i + 1]; y++){
                printf("Krok %d procesu %d \n", y, getpid());
                sleep(1);
            }
            exit(getpid());
        }
    }

    for(int i = 0; i < args[0]; i++){
            printf("Macierzysty krok %d\n",i);
            sleep(1);
    }

    for(int i = 0; i < argc - 2; i++){
        pid = wait(&status);

        printf("Proces %d zostal zakonczony, status  %d \n", pid, WEXITSTATUS(status));
    }

    exit(0);

}
```

Terminal po wykonaniu programu

```
cyta@Piotr:~/Desktop/ProgamowanieWspolbierzne$ ./startFork 2 2 2 2
Macierzysty krok 0
Krok 0 procesu 16498
Krok 0 procesu 16499
Krok 0 procesu 16500
Macierzysty krok 1
Krok 1 procesu 16498
Krok 1 procesu 16499
Krok 1 procesu 16500
Proces 16498 zostal zakonczony, status  114
Proces 16499 zostal zakonczony, status  115
Proces 16500 zostal zakonczony, status  116
```

Zadanie nr 2

kod wykonywany przez proces macierzysty

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


void main(int argc, char **argv){

    int pid, status;
    int args[argc - 1];

    for(int i = 1; i < argc; i++){
        args[i - 1] = atoi(argv[i]);
    }

    for(int i = 0; i < argc - 2; i++){//jak wiele procesow chce utworzyc
        if((pid = fork()) == 0){
            char str[12];
            sprintf(str, "%d", getpid());

            execl("./potEx", "potEx", str, argv[i + 2], (char*) NULL);


        }

    }

    for(int i = 0; i < atoi(argv[1]); i++){
            printf("Macierzysty krok %d\n",i);
            sleep(1);
    }

    for(int i = 0; i < argc - 2; i++){

        pid = wait(&status);

        printf("Proces %d zostal zakonczony, status  %d \n", pid, WEXITSTATUS(status));

    }


}
```

kod wykonywany przez procesy potomne

potEx.c > ...

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


int main(int argc, char const *argv[])
{

    for(int i = 0; i < atoi(argv[2]); i++){
        printf("Proces potomny nr. %d, krok nr. %d \n", atoi(argv[1]), i);
        sleep(1);
    }
    exit(0);
}

```

Terminal po wykonaniu programu

```
cyta@Piotr:~/Desktop/ProgamowanieWspolbierzne$ ./startEx 2 4 5 3
Macierzysty krok 0
Proces potomny nr. 10028, krok nr. 0
Proces potomny nr. 10029, krok nr. 0
Proces potomny nr. 10027, krok nr. 0
Macierzysty krok 1
Proces potomny nr. 10028, krok nr. 1
Proces potomny nr. 10029, krok nr. 1
Proces potomny nr. 10027, krok nr. 1
Proces potomny nr. 10028, krok nr. 2
Proces potomny nr. 10029, krok nr. 2
Proces potomny nr. 10027, krok nr. 2
Proces potomny nr. 10028, krok nr. 3
Proces potomny nr. 10027, krok nr. 3
Proces 10029 zostal zakonczony, status  0
Proces potomny nr. 10028, krok nr. 4
Proces 10027 zostal zakonczony, status  0
Proces 10028 zostal zakonczony, status  0
```

Zadanie nr 3

Kod wykonywany przez proces macierzysty

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void main(int argc, char **argv){

    int pid, status;
    int args[argc - 1];

    for(int i = 1; i < argc; i++){
        args[i - 1] = atoi(argv[i]);
    }


    for(int i = 0; i < argc - 2; i++){//jak wiele procesow chce utworzyc
        if((pid = fork()) == 0){
            char buf[80];

            char str[12];
            sprintf(str, "%d", getpid());

            sprintf(buf,"./potEx %s %s", str, argv[i + 2]);

            system(buf);
            _exit(i + 2);

        }

    }

    for(int i = 0; i < args[0]; i++){
            printf("Macierzysty krok %d\n",i);
            sleep(1);
    }

    for(int i = 0; i < argc - 2; i++){

        pid = wait(&status);

        printf("Proces %d zostal zakonczony, status  %d \n", pid, WEXITSTATUS(status));

    }
}
```

Kod wykonywany przez procesy potomne

potEx.c > ...

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


int main(int argc, char const *argv[])
{

    for(int i = 0; i < atoi(argv[2]); i++){
        printf("Proces potomny nr. %d, krok nr. %d \n", atoi(argv[1]), i);
        sleep(1);
    }
    exit(0);
}

```

Terminal po wykonaniu programu