

Programowanie współbieżne

Laboratoria 2

Piotr Stachnio 241268

18.03.2020

grupa zajęciowa nr. 2

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

Zadanie nr.1

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4
5
6  void main(int argc, char **argv){
7
8      int pid, status;
9      int args[argc - 1];
10
11     for(int i = 1; i < argc; i++){
12         args[i - 1] = atoi(argv[i]);
13     }
14
15     for(int i = 0; i < argc - 2; i++){//jak wiele procesow chce utworzyc
16         if((pid = fork()) == 0){
17
18             for(int y = 0; y < args[i + 1]; y++){
19                 printf("Krok %d procesu %d \n", y, getpid());
20                 sleep(1);
21             }
22             exit(getpid());
23         }
24     }
25
26     for(int i = 0; i < args[0]; i++){
27         printf("Macierzysty krok %d\n", i);
28         sleep(1);
29     }
30
31     for(int i = 0; i < argc - 2; i++){
32         pid = wait(&status);
33
34         printf("Proces %d zostal zakonczony, status %d \n", pid, WEXITSTATUS(status));
35     }
36
37     exit(0);
38
39 }
```

Terminal po wykonaniu programu

```
cyta@Piotr:~/Desktop/ProgramowanieWspolbierzne$ ./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

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4
5
6  void main(int argc, char **argv){
7
8      int pid, status;
9      int args[argc - 1];
10
11     for(int i = 1; i < argc; i++){
12         args[i - 1] = atoi(argv[i]);
13     }
14
15     for(int i = 0; i < argc - 2; i++){//jak wiele procesow chce utworzyc
16         if((pid = fork()) == 0){
17             char str[12];
18             sprintf(str, "%d", getpid());
19
20             execl("./potEx", "potEx", str, argv[i + 2], (char*) NULL);
21         }
22     }
23
24 }
25
26
27 for(int i = 0; i < atoi(argv[1]); i++){
28     printf("Macierzysty krok %d\n", i);
29     sleep(1);
30 }
31
32 for(int i = 0; i < argc - 2; i++){
33
34     pid = wait(&status);
35
36     printf("Proces %d zostal zakonczony, status %d \n", pid, WEXITSTATUS(status));
37 }
38
39
40
41 }
```

kod wykonywany przez procesy potomne

```
C potEx.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4
5
6  int main(int argc, char const *argv[])
7  {
8
9      for(int i = 0; i < atoi(argv[2]); i++){
10         printf("Proces potomny nr. %d, krok nr. %d \n", atoi(argv[1]), i);
11         sleep(1);
12     }
13     exit(0);
14 }
15
```

Terminal po wykonaniu programu

```
cyta@Piotr:~/Desktop/ProgarnowanieWspolbierzne$ ./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

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4
5  void main(int argc, char **argv){
6
7      int pid, status;
8      int args[argc - 1];
9
10     for(int i = 1; i < argc; i++){
11         args[i - 1] = atoi(argv[i]);
12     }
13
14
15     for(int i = 0; i < argc - 2; i++){//jak wiele procesow chce utworzyc
16         if((pid = fork()) == 0){
17             char buf[80];
18
19             char str[12];
20             sprintf(str, "%d", getpid());
21
22             sprintf(buf, "./potEx %s %s", str, argv[i + 2]);
23
24             system(buf);
25             _exit(i + 2);
26         }
27     }
28
29 }
30
31 for(int i = 0; i < args[0]; i++){
32     printf("Macierzysty krok %d\n", i);
33     sleep(1);
34 }
35
36 for(int i = 0; i < argc - 2; i++){
37
38     pid = wait(&status);
39
40     printf("Proces %d zostal zakonczony, status %d \n", pid, WEXITSTATUS(status));
41 }
42
43 }
```

Kod wykonywany przez procesy potomne

```
potEx.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4
5
6  int main(int argc, char const *argv[])
7  {
8
9      for(int i = 0; i < atoi(argv[2]); i++){
10         printf("Proces potomny nr. %d, krok nr. %d \n", atoi(argv[1]), i);
11         sleep(1);
12     }
13     exit(0);
14 }
15
```

Terminal po wykonaniu programu

```
cyta@Piotr: ~/Desktop/ProgamowanieWspolbierzne
cyta@Piotr:~/Desktop/ProgamowanieWspolbierzne$ ./startSystem 2 4 3
Macierzysty krok 0
Proces potomny nr. 9286, krok nr. 0
Proces potomny nr. 9287, krok nr. 0
Macierzysty krok 1
Proces potomny nr. 9286, krok nr. 1
Proces potomny nr. 9287, krok nr. 1
Proces potomny nr. 9286, krok nr. 2
Proces potomny nr. 9287, krok nr. 2
Proces potomny nr. 9286, krok nr. 3
Proces 9287 zostal zakonczony, status 3
Proces 9286 zostal zakonczony, status 2
cyta@Piotr:~/Desktop/ProgamowanieWspolbierzne$
```