МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський Політехнічний Інститут»

Кафедра Стратегічного управління

ЗВІТ

з лабораторної роботи № 3

з дисципліни «Операційні системи»

на тему «Передача параметрів командної строки в программу. Робота з процесами. Отримання системної інформації»

Перевірила: старший викладач

Вільхівська О. В.

Виконав: ст. гр. КН-27

Харків, 2018

Л**абораторная работа №3**

**Тема: Передача параметров командной строки в программу. Работа процессами. Получение системной информации.**

**Задания**

**Задание 1:** Разработать два приложения для Linux, первое из которых:

а) при задании ключа -w имя-исполняемого-файла - запускает заданное

приложение (с поиском его в системном пути, заданном переменной

окружения PATH) - ожидает завершения приложения и выводит

сообщение о завершении на экран вместе с кодом возврата

б) при задании ключа -f имя-исполняемого-файла - запускает заданное

приложение (с поиском его в системном пути, заданном переменной

окружения PATH) - немедленно продолжает выполнение (асинхронный

запуск)

Второе приложение должно выводить на экран:

значения аргументов командной строки; идентификаторы: процесса (pid), сессии (sid), группы процессов

(pgid)

и процесса-предка (ppid)

имя пользователя, под которым процесс запущен и его группу

(полученные из файлов /etc/passwd и /etc/group)

**Задание 2:** Вариант No2. Разработать две программы. Первая вычисляет число Фибоначчи по номеру, введенному пользователю, и формуле Fi = Fi–1 + Fi–2, F0 = F1=1 и выводит его на экран. Вторая программа запускает первую в

качестве вновь созданного процесса.

**Ход работы**

**Задание 1.**

Уловие а (ожидание завершения работы предка). Код программы:

|  |
| --- |
| #include <cstdlib>  #include <iostream>  #include <getopt.h>  #include <stdlib.h>  #include <stdio.h>  #include <unistd.h>  #include <sys/types.h>  #include <pwd.h>  #include <grp.h>  #include <sys/time.h>  #include <sys/resource.h>  //#include <sys/types.h>  #include <sys/wait.h>  #include <errno.h>  int main(int argc, char \*argv[]) {  int c;  while (1) {  c = getopt(argc, argv, "w:");  if (c == -1) break;  switch (c) {  case 'w':  printf ("задано -f с параметром: %s\n", optarg);  pid\_t pid;  int status;  if ((pid=fork()) == -1){  printf ("fork error\n");  exit(-1);  }  if (pid == 0) { /\* Code executed by child \*/  printf("Child PID is %ld\n", (long) getpid());  char \*argv[2];  argv[0] = optarg;  argv[1] = NULL;  if (execvp (optarg, argv) == -1) {  printf ("error executing the program: %s\n", optarg);  exit (-1);  }  exit(777);  } else { /\* Code executed by parent \*/  waitpid (pid, &status, 0);  if (WIFEXITED(status))  printf("потомок завершился с кодом %d\n", WEXITSTATUS(status));  }  break;  }  }  } |

Пример работы:

|  |
| --- |
| reacher@reacher-X550CC:~/NetBeansProjects/lab3$ g++ main.cpp  reacher@reacher-X550CC:~/NetBeansProjects/lab3$ ./a.out -w ls  задано -f с параметром: ls  Child PID is 27269  a.out build child.cpp dist main.cpp Makefile nbproject  потомок завершился с кодом 0 |

Условие б (без ожидания завершения работы предка). Код программы:

|  |
| --- |
| #include <cstdlib>  #include <iostream>  #include <getopt.h>  #include <stdlib.h>  #include <stdio.h>  #include <unistd.h>  #include <sys/types.h>  #include <pwd.h>  #include <grp.h>  #include <sys/time.h>  #include <sys/resource.h>  //#include <sys/types.h>  #include <sys/wait.h>  #include <errno.h>  using namespace std;  /\*  \*  \*/  extern char \*\*environ;  int task\_1 (int argc, char \*argv[]) {  }  int main(int argc, char \*argv[]) {  int c;  while (1) {  c = getopt(argc, argv, "w:");  if (c == -1) break;  switch (c) {  case 'w':  printf ("задано -f с параметром: %s\n", optarg);  pid\_t pid;  int status;  if ((pid=fork()) == -1){  printf ("fork error\n");  exit(-1);  }  if (pid == 0) { /\* Code executed by child \*/  printf("Child PID is %ld\n", (long) getpid());  char \*argv[2];  argv[0] = optarg;  argv[1] = NULL;  if (execvp (optarg, argv) == -1) {  printf ("error executing the program: %s\n", optarg);  exit (-1);  }  exit(777);  } else { /\* Code executed by parent \*/  printf("потомок еще не завершился, а предок продолжает работу\n");  waitpid (pid, &status, 0);  if (WIFEXITED(status))  printf("потомок завершился с кодом %d\n", WEXITSTATUS(status));  }  break;  }  }  } |

Пример работы:

|  |
| --- |
| reacher@reacher-X550CC:~/NetBeansProjects/lab3$ g++ task\_2.cpp  reacher@reacher-X550CC:~/NetBeansProjects/lab3$ ./a.out -w ls  задано -f с параметром: ls  потомок еще не завершился, а предок продолжает работу  Child PID is 834  a.out build dist main.cpp Makefile nbproject task\_2.cpp  потомок завершился с кодом 0  reacher@reacher-X550CC:~/NetBeansProjects/lab3$ |

**Задание 2.**

Код программы файл fibonacci.cpp

|  |
| --- |
| #include <stdio.h>  #include <cstdlib>  #include <iostream>  using namespace std;  int main(int argc, char\*\* argv)  {  int n, c, first = 0, second = 1, next;    cout << "Enter the number of terms of Fibonacci series you want" << endl;  cin >> n;    cout << "First " << n << " terms of Fibonacci series are :- " << endl;    for ( c = 0 ; c < n ; c++ )  {  if ( c <= 1 )  next = c;  else  {  next = first + second;  first = second;  second = next;  }  cout << next << endl;  }    return 0;  } |

Код программы файл loader.cpp

|  |
| --- |
| #include <cstdlib>  #include <iostream>  #include <getopt.h>  #include <stdlib.h> // Env. variables  #include <stdio.h> // Env. variables  #include <unistd.h> // IDs  #include <sys/types.h> // IDs  #include <pwd.h> // etc/password  #include <grp.h> // group info  #include <sys/time.h> // Reousrces limitation  #include <sys/resource.h> // Reousrces limitation  //#include <sys/types.h> // For fork  #include <sys/wait.h> // Wating for the descender 's finish  #include <errno.h>  using namespace std;  /\*  \*  \*/  int main(int argc, char\*\* argv) {  int status;  pid\_t pid;  if ((pid=fork()) == -1){  printf ("fork error\n");  exit(-1);  }  if (pid == 0) { /\* Code executed by child \*/  printf("Child PID is %ld\n", (long) getpid());  char \*prog = (char\*)"./a.out";  char \*args[] = { (char\*)"./a.out", NULL};  if (execvp (prog, args) == -1) {  printf ("error executing the program: %s\n", optarg);  exit (-1);  }  exit(777);  } else { /\* Code executed by parent \*/  waitpid (pid, &status, 0);  if (WIFEXITED(status))  printf("потомок завершился с кодом %d\n", WEXITSTATUS(status));  }  return 0;  } |

Пример работы

|  |
| --- |
| reacher@reacher-X550CC:~/NetBeansProjects/CppApplication\_2$ g++ fibonacci.cpp  reacher@reacher-X550CC:~/NetBeansProjects/CppApplication\_2$ ./loader  Child PID is 31641  error executing the program: (null)  потомок завершился с кодом 255  reacher@reacher-X550CC:~/NetBeansProjects/CppApplication\_2$ g++ loader.cpp -o loader  reacher@reacher-X550CC:~/NetBeansProjects/CppApplication\_2$ ./loader  Child PID is 32085  Enter the number of terms of Fibonacci series you want  10  First 10 terms of Fibonacci series are :-  0  1  1  2  3  5  8  13  21  34  потомок завершился с кодом 0  reacher@reacher-X550CC:~/NetBeansProjects/CppApplication\_2$ |

**Выводы**

В ходе выполнения лабораторной работы я ознакомился с функцией fork(), которая создает новый процесс - копию родителя. Процессы получают разные коды возврата от процедуры fork() (0 для потомка и значение PID потомка для предка). Таким образом их можно различать и ставить процесс-потомок на ожидание с помощью wait() или waitpid(). Также узнал о зомби-процессах - процессах потомках, которые уже завершились, но информация об их состоянии еще не была удалена из таблицы процессов (в частном случае - еще не вызвана команда wait() процесса-предка).