Лабораторная работа 03

ОС (**Exp.: 06.10.2024)**

Это достаточно простая работа, поэтому рекомендую выполнить ее полностью самостоятельно, образцы кода можно найти в приложении к этому документу и в старых лекциях Смелова в этой же папке.

Задание 01

1. Разработайте консольное Windows-приложение **OS03_01** на языке C++, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.
2. Продемонстрируйте информацию о процессе **OS03_01** с помощью утилит **Task Manager**, **tasklist**, **PowerShell ISE** и **Performance Monitor**.

Задание 02

3. Разработайте консольное Windows-приложение **OS03_02** на языке C++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
4. Приложение **OS03_02** должно создавать два дочерних процесса **OS03_02_1** и **OS03_02_2.**
5. Процесс **OS03_02_1** - консольное Windows-приложение, выполняющее цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
6. Процесс **OS03_02_2** - консольное Windows-приложение выполняющее цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
7. Продемонстрируйте информацию о процессах **OS03_02, OS03_02_1** и **OS03_02_2** с помощью утилит **Task Manager, tasklist, PowerShell ISE** и **Performance Monitor**.

Задание 03.

8. Разработайте консольное Windows-приложение **OS03_03** на языке C++, выводящее на консоль перечень выполняющихся процессов в данный момент в OS.
9. Запустите приложение **OS03_02** и продемонстрируйте с помощью приложения **OS03_03** в перечне процессов **OS03_02, OS03_02_1, OS03_02_2** и **OS03_03.**

Задание 04

10. Разработайте консольное Linux-приложение **OS03_04** на языке C, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.
11. Продемонстрируйте информацию о процессе **OS03_04** с помощью файловой системы **/proc.**
12. Продемонстрируйте информацию о процессе **OS03_04** с помощью утилиты **ps.**

Задание 05

13. Разработайте консольное Linux-приложение **OS03_05** на языке C, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
14. Приложение **OS03_05** должно создавать один дочерний процесс **OS03_05_1** с помощью системного вызова **fork.** Процесс **OS03_05_1** в этом случае не является отдельным модулем, а встроен (fork) в программный модуль **OS03_05.**
15. Процесс **OS03_05_1** - консольное Linux-приложение, выполняющее цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
16. Продемонстрируйте информацию о процессах **OS03_05** и **OS03_05_1** с помощью файловой системы **/proc.**
17. Продемонстрируйте информацию о процессах **OS03_05** и **OS03_05_1** с помощью утилиты **ps.**

Задание 06

18. Разработайте консольное Linux-приложение **OS03_06** на языке C, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
19. Приложение **OS03_06** должно создавать один дочерний процесс **OS03_05_1** (отдельный модуль) с помощью системного вызова **system.**
20. Продемонстрируйте информацию о процессах **OS03_06** и **OS03_05_1** с помощью файловой системы **/proc.**
21. Продемонстрируйте информацию о процессах **OS03_06** и **OS03_05-1** с помощью утилиты **ps.**

Задание 07

22. Разработайте консольное Linux-приложение **OS03_07** на языке C, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса
23. Приложение **OS03_07** должно создавать один дочерний процесс **OS03_05_1** (отдельный модуль) с помощью системного вызова **exec.**
24. Продемонстрируйте информацию о процессах **OS03_07** и **OS03_05_1** с помощью файловой системы **/proc.**
25. Продемонстрируйте информацию о процессах **OS03_07** и **OS03_05-1** с помощью утилиты **ps.**
26. Продемонстрируйте разницу системных вызовов **system** и **exec.**

Задание 08. Ответьте на следующие вопросы

27. Что такое процесс?
28. Что такое контекст процесса?
29. Что такое родительский и дочерний процесс?
30. Что такое процесс инициализации OS?
31. Перечислите области памяти процесса и поясните их назначение.
32. Чем отличаются системные процессы от пользовательских?
33. Что такое Windows-сервисы, Linux-демоны?
34. С помощью каких системных вызовов можно создать дочерний процесс в Windows? Поясните разницу.
35. С помощью каких системных вызовов можно создать дочерний процесс в Linux? Поясните разницу.
36. Какие потоки данных доступны любому процессу автоматически?
37. Поясните назначение системного вызова WaitForSingleObject в Windows-приложении.
38. Поясните назначение системного вызова wait в Linux-приложении.
39. Дайте развернутое определение процесса OS.

Приложения

Создание дочернего процесса в Windows

```
STARTUPINFO si;
PROCESS_INFORMATION pi;
ZeroMemory(&si, sizeof(si));
si.cb = sizeof(si);
ZeroMemory(&pi, sizeof(pi));

BOOL bRes = CreateProcessW(L"C:\\Windows\\notepad.exe", NULL,
    NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi);
if (!bRes)
    printf("Error %d\n", GetLastError());

WaitForSingleObject(pi.hProcess, INFINITE);

// Close process and thread handles.
CloseHandle(pi.hProcess);
CloseHandle(pi.hThread);
```

Отображение только рабочих процессов с именами, начинающимися на OS, в Linux

```
ps -eF | grep [O]S
```

Здесь квадратные скобки препятствуют попаданию в выходной список самой выполняемой команды `ps`

Создание дочернего процесса в Linux

```
int main() {
   int pid = fork();
   switch(pid) {
      case -1:
         perror("fork");
         return -1;
      case 0: // Child
         printf("my pid = %i, returned pid = %i\n", getpid(), pid);
         break;
      default: // Parent
         printf("my pid = %i, returned pid = %i\n", getpid(), pid);
         break;
   }
   return 0;
}
```