Лабораторная работа 03

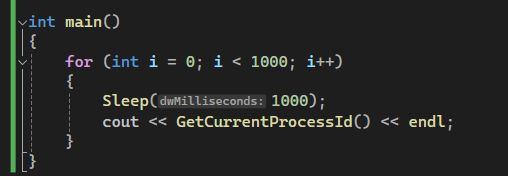
# OC (Exp.: 06.10.2024)

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 С++, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.

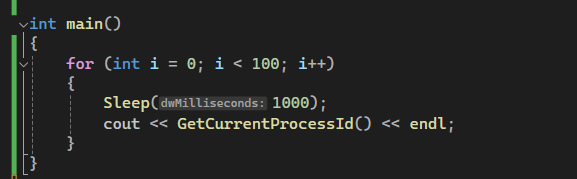


1. Продемонстрируйте информацию о процессе **OS03\_01** с помощью утилит **Task Manager**, **tasklist, PowerShell ISE** и **Performance Monitor**.

|  |
| --- |
| 1. Task Manager:  Диспетчер задач -> Подробности -> OS03\_01.exe    2. Tasklist:  tasklist /FI "IMAGENAME eq OS03\_01.exe" ИЛИ tasklist FI "pid eq PID" или tasklist | find “OS03\_01”    3. PowerShell ISE:  Get-Process OS03\_01    4. Performance Monitor: |

# Задание 02

1. Разработайте консольное Windows-приложение **OS03\_02** на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.



1. Приложение **OS03\_02** должно создавать два дочерних процесса **OS03\_02\_1** и **OS03\_02\_2.**
2. Процесс **OS03\_02\_1** - консольное Windows-приложение, выполняющее цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
3. Процесс **OS03\_02\_2** - консольное Windows-приложение выполняющее цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
4. Продемонстрируйте информацию о процессах **OS03\_02, OS03\_02\_1** и **OS03\_02\_2** с помощью утилит **Task Manager**, **tasklist, PowerShell ISE** и **Performance Monitor**.

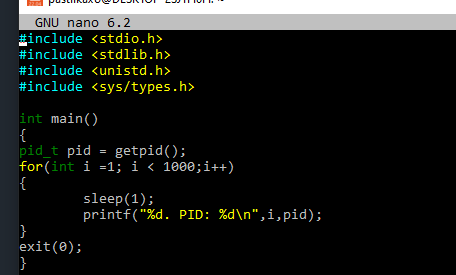
|  |
| --- |
| 1) Task Manager      2) tasklist    2) **PowerShell ISE**    3)Perfomance Monitor |

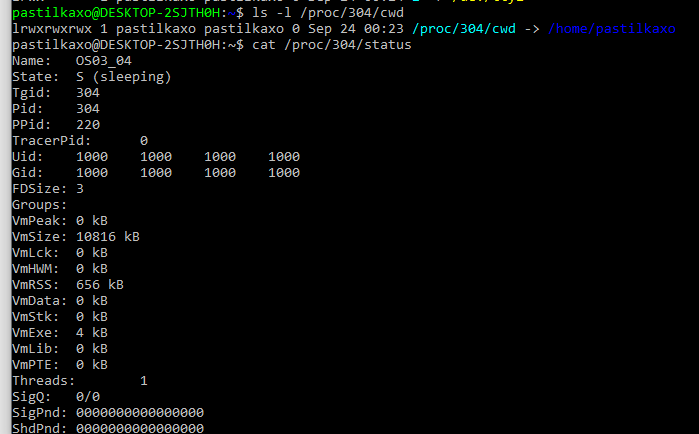
# Задание 03.

1. Разработайте консольное Windows-приложение **OS03\_03** на языке С++, выводящее на консоль перечень выполняющихся процессов в данный момент в OS.
2. 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

1. Разработайте консольное Linux-приложение **OS03\_04** на языке С, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.
2. 
3. Продемонстрируйте информацию о процессе **OS03\_04** с помощью файловой системы **/proc**.

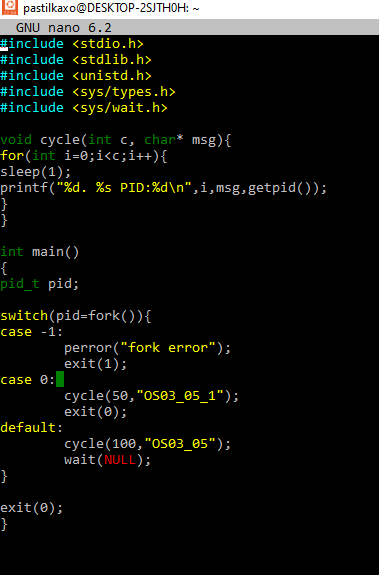


1. Продемонстрируйте информацию о процессе **OS03\_04** с помощью утилиты **ps**.



# Задание 05

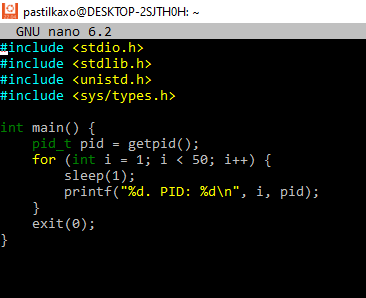
1. Разработайте консольное Linux-приложение **OS03\_05** на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.



1. 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**.
2. Процесс **OS03\_05\_1** - консольное Linux-приложение,

выполняющее цикл 50 итераций с временной задержкой в

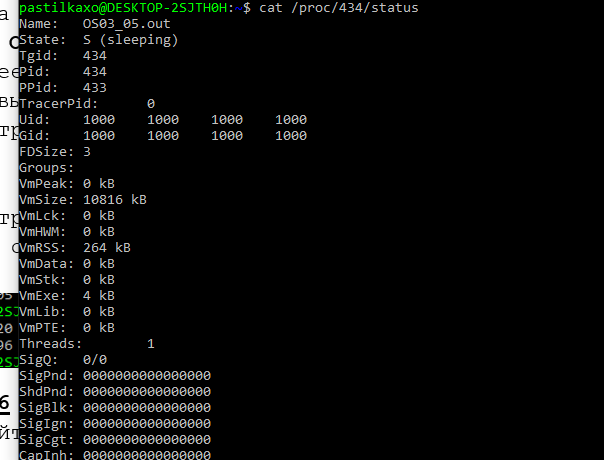
1 сек. с выводом на консоль идентификатора процесса.

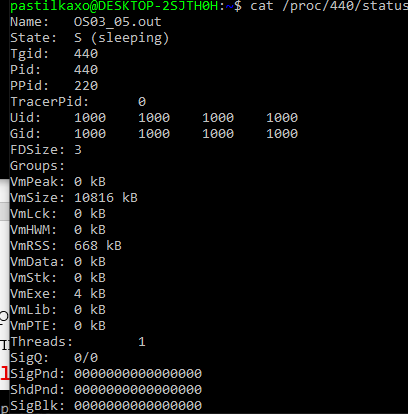


1. Продемонстрируйте информацию о процессах **OS03\_05** и

**OS03\_05\_1** с помощью файловой системы **/proc**.

**cat /proc/pid/status**

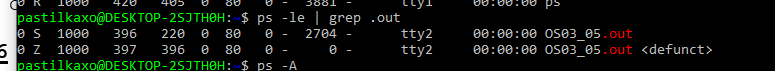
Child:  


Parent:  


1. Продемонстрируйте информацию о процессах **OS03\_05** и

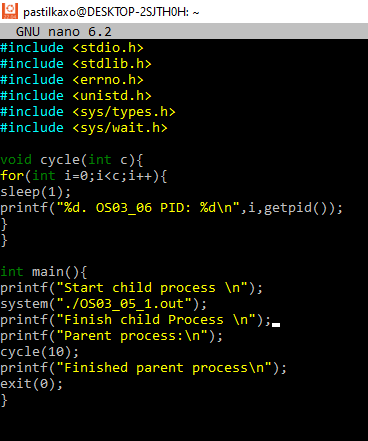
**OS03\_05\_1** с помощью утилиты **ps**.

**ps -le | grep .out**



# Задание 06

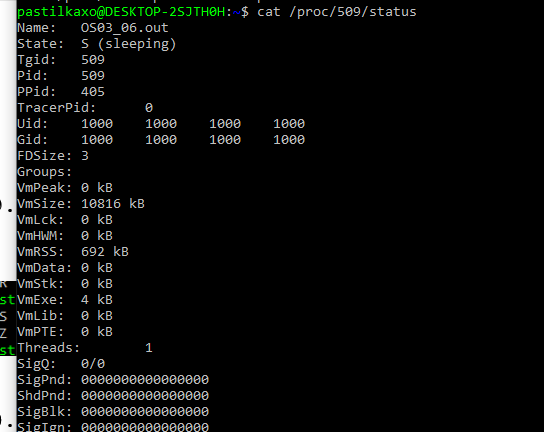
1. Разработайте консольное Linux-приложение **OS03\_06** на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.



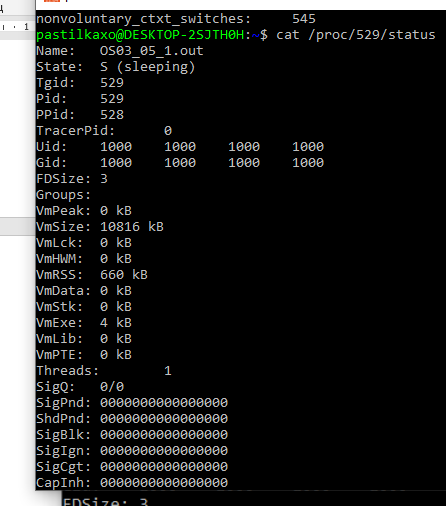
1. Приложение **OS03\_06** должно создавать один дочерний процесс **OS03\_05\_1** (отдельный модуль) с помощью системного вызова **system.**
2. Продемонстрируйте информацию о процессах **OS03\_06** и

**OS03\_05\_1** с помощью файловой системы **/proc**

Parent:



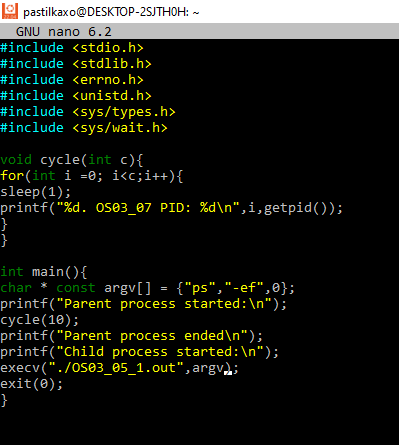
Child:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 21. Продемонстрируйте информацию | о | процессах | **OS03\_06** | и |
| **OS03\_05-1** с помощью утилиты | **ps**. |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

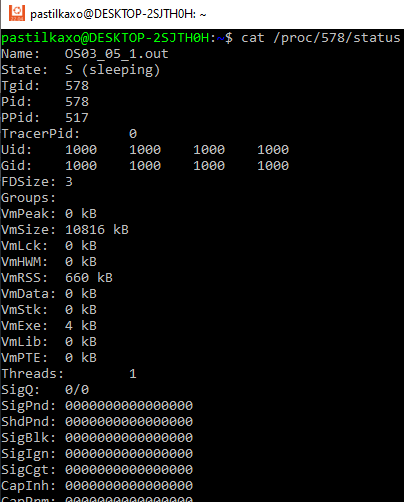
# Задание 07

1. Разработайте консольное Linux-приложение **OS03\_07** на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса



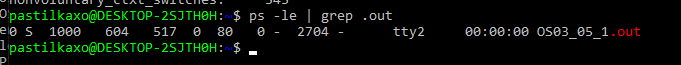
1. Приложение **OS03\_07** должно создавать один дочерний процесс **OS03\_05\_1** (отдельный модуль) с помощью системного вызова **exec.**
2. Продемонстрируйте информацию о процессах **OS03\_07** и

**OS03\_05\_1** с помощью файловой системы **/proc**.



1. Продемонстрируйте информацию о процессах **OS03\_07** и

**OS03\_05-1** с помощью утилиты **ps**.



1. Продемонстрируйте разницу системных вызовов **system** и

# exec.

# exec:

# system:

# 

system() - если exec полностью заменяет родительский процесс и никогда в него не возвращается, то system разветвляется, запускает дочерний процесс и возвращает его статус. то есть system запускает дочерний и ставит в ожидание родительский, а после завершения дочернего продолжает работу с родителем. функционал system можно повторить с помощью fork exec wait

exec() - заменяет текущий процесс другим. загружает новую программу в адресное пространство текущего процесса. тк новый процесс не создается, то pid не меняется. новый процесс заменяет данные, кучу, стек и машинный код текущего процесса. он загружает новый процесс в текущее адресное пространство и выполняет его из точки входа. управление никогда не возвращается к исходному процессу, если нет ошибки 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? |

1. Перечислите области памяти процесса и поясните их назначение.
2. Чем отличаются системные процессы от пользовательских?
3. Что такое Windows-сервисы, Linux-демоны?
4. С помощью каких системных вызовов можно создать дочерний процесс в Windows? Поясните разницу.
5. С помощью каких системных вызовов можно создать дочерний процесс в Linux? Поясните разницу.
6. Какие потоки данных доступны любому процессу автоматически?
7. Поясните назначение системного вызова

WaitForSingleObject в Windows-приложении.

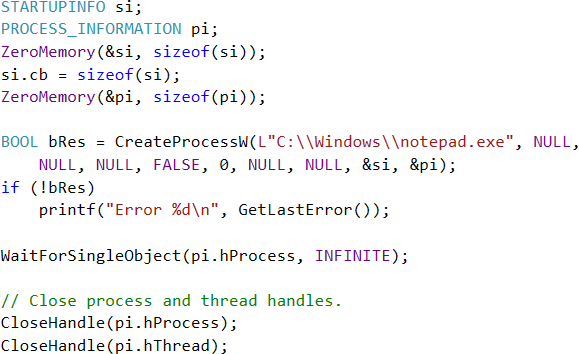
1. Поясните назначение системного вызова wait в Linux-

приложении.

1. Дайте развернутое определение процесса OS.

**Приложения**

# Создание дочернего процесса в Windows



**Отображение только рабочих процессов c 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;

}