Семинар #8: Процессы. Практика.

Как сдавать задачи

Для сдачи ДЗ вам нужно создать репозиторий на GitLab (если он ещё не создан) под названием `devtools-homework`. Структура репозитория должна иметь вид:

```
├── seminar8_processes/
│   ├── 01.sh
│   ├── 02.sh
│   └── ...
└── ...
```

Для каждой задачи, если в самой задаче не сказано иное, нужно создать 1 скрипт с расширением `.sh` и шебангом в начале скрипта. Если задача делится на подзадачи нужно, если в самой задаче не сказано иное, создать скрипт для каждой подзадачи. Названия файлов решений для всех задач/подзадач должны начинаться с номера задачи, например `01.sh` или `04b.sh`, даже если в условии задачи используется другое имя для скрипта.

Задача 1. Запуск в фоновом режиме

(a) Создадим скрипт `counter.sh` который будет печатать на экран возрастающие целые числа с шагом в пол секунды.

```
#!/bin/bash
i=0
while true; do
    echo "$i"
    sleep 0.5
    ((i += 1))
done
```

Напишите или скачайте этот скрипт с репозитория и дайте ему права на исполнение.

(b) Запустите скрипт:

```
$ ./counter.sh
```

В результате на экран будут печататься числа. Bash будет занят выводом чисел на экран, поэтому в данной оболочке мы ничего сделать не сможем, пока не завершим программу. Завершите исполнение программы, используя `Ctrl-C`.

(c) Запустите скрипт в фоновом режиме:

```
$ ./counter.sh &
```

Bash запустит программу в фоновом режиме. В этом случае bash будет свободен и вы сможете выполнять другие команды. Правда, так как `counter.sh` всё ещё выводит числа на экран, то работа с оболочкой может быть затруднена. Завершить программу, используя `Ctrl-C` в данном случае невозможно. Чтобы завершить процесс, его нужно сначала перевести из фонового (background) режима в передний (foreground) режим, используя команду `fg` и после этого завершить с помощью `Ctrl-C`.

(d) Запустите скрипт с выводом в файл:

```
$ ./counter.sh > a.txt
```

В результате в файл `a.txt` будут записываться числа. Bash будет занят записью чисел в файл, поэтому в данной оболочке мы ничего сделать не сможем. Но можно проверить, что числа печатаются в файл, используя другой терминал. Откройте другой терминал, зайдите в ту же директорию и проверьте, что в файл `a.txt` в данный момент производится запись, несколько раз вызвав `cat`. Также можно использовать команду:

```
$ watch -n 2 cat a.txt
```

которая будет автоматически повторять команду `cat a.txt` каждые 2 секунды. Но, так как запись происходит в конец файла, то скорей всего лучше использовать команду `tail` вместо `cat`.

```
$ watch -n 2 tail a.txt
```

Задача по выводу последних строк изменяемого файла используется настолько часто, что команде `tail` дали специальную опцию для этого случая:

```
$ tail -f a.txt
```

Которая выводит последние строки файла, после изменения этого файла.

Закройте второй терминал, вернитесь на первый и завершите процесс `counter.sh`.

(e) Запустите скрипт с выводом в файл в фоновом режиме:

```
$ ./counter.sh > a.txt &
```

В результате в файл `a.txt` будут записываться числа. В этом случае bash будет свободен и вы сможете выполнять другие команды. Проверьте, что программа `counter` работает и в файл записываются числа, используя команду `tail -f`. Чтобы завершить процесс, его нужно перевести из фонового (background) режима в передний (foreground) режим, используя команду `fg` и после этого завершить с помощью `Ctrl-C`.

(f) Запустите скрипт с выводом в файл в фоновом режиме:

```
$ ./counter.sh > a.txt &
```

Выполните команду `ps`, которая покажет все программы, запущенные в данном терминале. Скорей всего на экран выведется информация о процессах `bash` (оболочка), `counter.sh`, `sleep` (запушенный внутри `counter.sh`) и только что запущенный `ps`. Для каждого процесса, будет выведен его идентификатор PID (*process identifier*). Используйте полученный PID процесса `counter.sh` и команду `kill`, чтобы завершить процесс. Проверьте, что процесс завершился и запись в `a.txt` больше не производится. Ещё раз выполните команду `ps` и убедитесь, что процессы `counter.sh` и `sleep` отсутствуют в выводе.

Задача 2. Приостановка процесса

(a) **Простая приостановка**

- Запустите скрипт `counter.sh`:

  ```
  $ ./counter.sh
  ```

- Нажмите комбинацию клавиш `Ctrl-Z`, чтобы приостановить процесс. Теперь процесс приостановился и ничего не выводит на экран.
- Для возобновления работы процесса на переднем плане используйте команду `fg` (*foreground process*).
- Завершите процесс, используя `Ctrl-C`.

(b) **Передний и фоновый режимы**

- Запустите скрипт `counter.sh` с записью в файл:

  ```
  $ ./counter.sh > a.txt
  ```

- Откройте новый терминал и запустите на нём просмотр конца файла `a.txt`, используя `tail -F a.txt`. Убедитесь, что в файл происходит запись. Не закрывайте этот терминал, на нём мы будем проверять, что наш процесс работает.
- Перейдите на основной терминал. Нажмите комбинацию клавиш `Ctrl-Z`, чтобы приостановить процесс. Теперь процесс приостановился и ничего не будет записывать в файл. Убедитесь в этом на втором терминале.
- Используйте команду `fg`, чтобы возобновить работу процесса на переднем плане. Убедитесь, что процесс начал работу, проверив вывод команды `tail` на втором терминале.
- Снова приостановите процесс, используя `Ctrl-Z`. Используйте команду `bg`, чтобы возобновить процесс, но теперь в фоновом режиме. Убедитесь, что процесс начал работать, посмотрев вывод на другом терминале.
- Переведите процесс на передний план и завершите его.

(c) **Список задач**

– Завершите все предыдущие процессы и запустите скрипт три раза в фоновом режиме:

```
$ ./counter.sh > a.txt &
$ ./counter.sh > b.txt &
$ ./counter.sh > c.txt &
```

– Убедитесь, что все задачи работают в фоновом режиме, проверив запись в соответствующие файлы.

– Выполните команду `jobs`, чтобы посмотреть все фоновые или приостановленные задачи в данной оболочке. Символом `+` будет отображаться последняя добавленная в `jobs` задача, а символом `-` предпоследняя.

– Переведите задачу записи в `a.txt` на передний план, используя команду:

```
$ fg %N   # вместо N нужно подставить номер задачи в jobs
```

Остановите эту задачу с помощью `Ctrl-Z` и снова посмотрите на список задач.

– Теперь задача записи в `a.txt` приостановлена. Обратите внимание, что рядом с задачей записи в `a.txt` стоит символ `+`, так как эта последняя добавленная задача. Возобновите работу этой программы в фоновом режиме, используя:

```
$ bg %N   # вместо N нужно подставить номер задачи в jobs
```

или, так как эта задача является последней добавленной, то можно использовать `bg` без указания номера. В этом случае операция будет применена к последней добавленной задаче. Снова посмотрите на все задачи, используя `jobs`.

– Используйте команду

```
$ kill %N   # вместо N нужно подставить номер задачи в jobs
```

чтобы завершить задачу записи в файл `c.txt`. Снова посмотрите на все задачи.

– Используйте команду

```
$ kill -STOP %N   # вместо N нужно подставить номер задачи в jobs
```

чтобы остановить задачу записи в файл `b.txt`. Посмотрите на все задачи. Используйте `bg` или

```
$ kill -CONT %N   # вместо N нужно подставить номер задачи в jobs
```

чтобы возобновить приостановленную задачу.

Задача 3. Список процессов

- Напечатайте все процессы данного терминала, используя `ps`.
- Напечатайте все процессы, используя `ps -ef`.
- Напечатайте все процессы, показав информацию только о uid, pid, ppid, pcpu, pmem, time.
- Напечатайте все процессы, показав информацию только о uid, pid, ppid, pcpu, pmem, time и отсортировав по полю pmem.

Задача 4. Родословная

Напишите скрипт, который будет принимать через аргумент PID процесса и выводить рекурсивно PID его родителей и команды, с помощью которых были все процессы запущены. Последний родитель будет иметь имя `systemd` (или `/sbin/init`) и иметь PID равный 1.

Задача 5. Ловец сигналов

Напишите скрипт, который будет ловить сигналы и при поимке сигнала писать сообщение

```
I caught названиесигнала
```

Протестируйте скрипт, отправив ему сигналы `SIGINT`, `SIGTERM`, `SIGHUP`, `SIGTSTP`, `SIGCONT`, `SIGQUIT`.