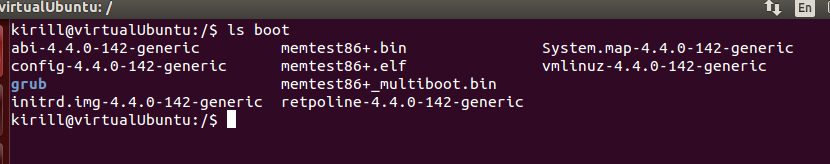
Лабораторная работа №7

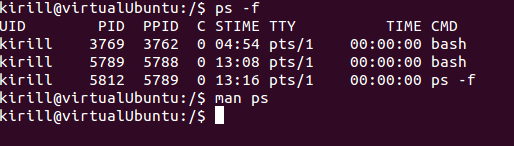
1. Войдем под пользователем, а не под root



1. Узнаем версию ядра vmlinuz, откроем каталог, содержащий ее

версия 4.4.0.142

1. Просмотрим все процессы



Pdi – идентификатор процесса

Ppdi идентификатор родительского процесса

Tty имя управляющего терминала – с которого запущен процесс

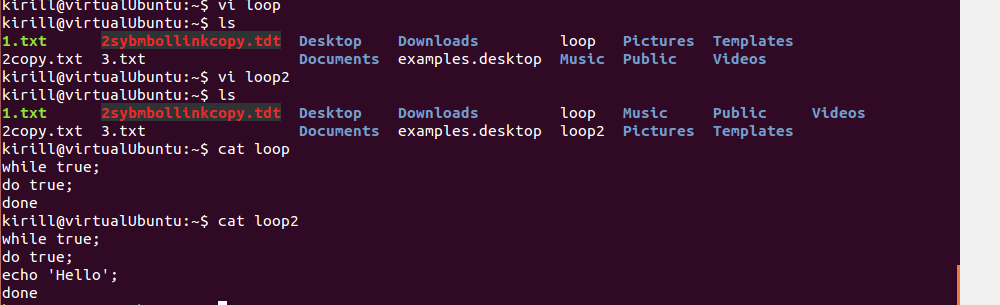
C – использование ресурсов процессора в процентах

CMD – команда которой был запущен процесс

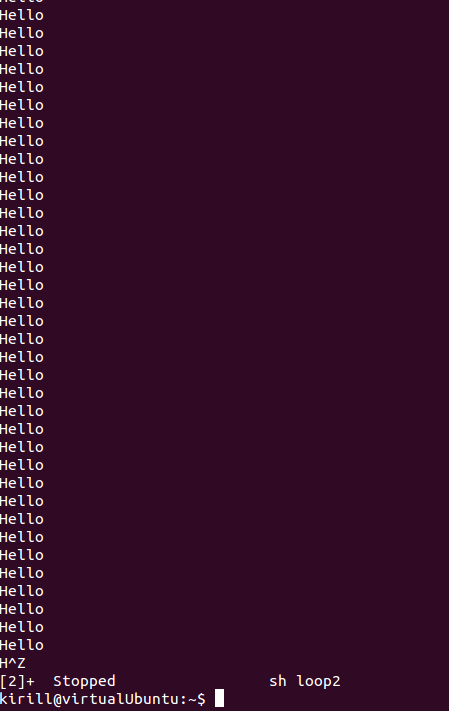
Time – время выполнения процесса

1. Создадим два сценария loop and loop 2.

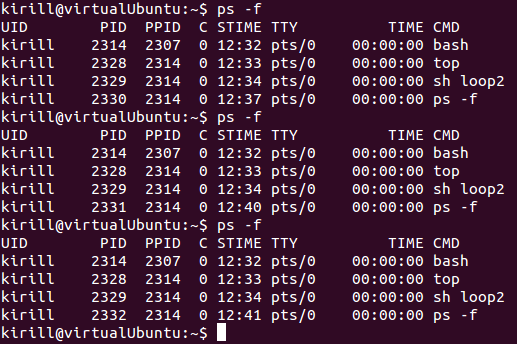
создадим два текстовых файла с помощью ред vi и напишем интерпретируемый код



1. Запустим процесс loop2 на переднем плане командой sh loop2 и посмотрим результат



1. Сразу же его остановим ctrl + z
2. Вызовем несколько раз ps –f и посмотрим, что изменилось



Видно, что наш остановленный процесс loop2 никак не изменяется

1. Убьем процесс командой kill -9 PID

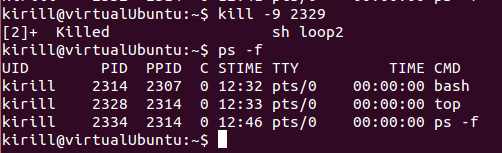
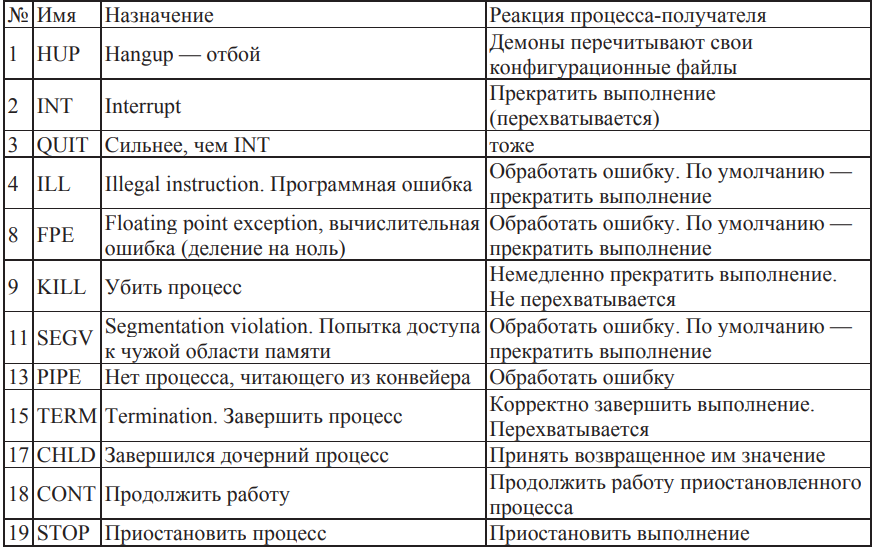
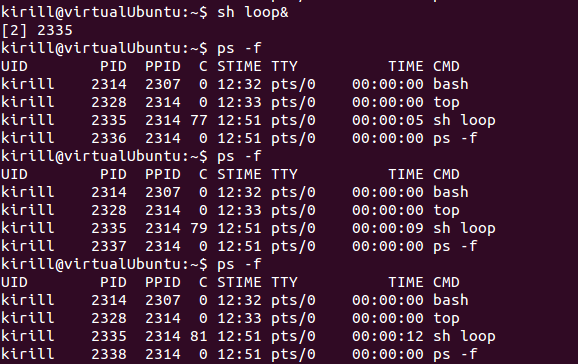


Таблица значений номера сигнала в lunix

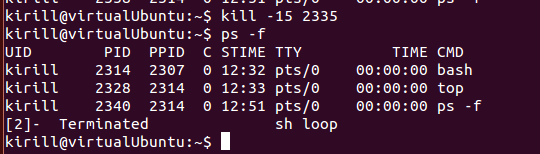


1. Запустим процесс sh loop в фоне с помощью sh loop& и посмотри несколько раз ps –f



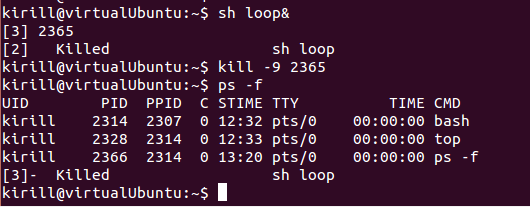
Как видим, количество используемых ресурсов процессора С постепенно возрастает, время выполнения процесса TIME тоже растет.

1. Завершим процесс loop командой kill -15 PID – Termination. После вызовем ps



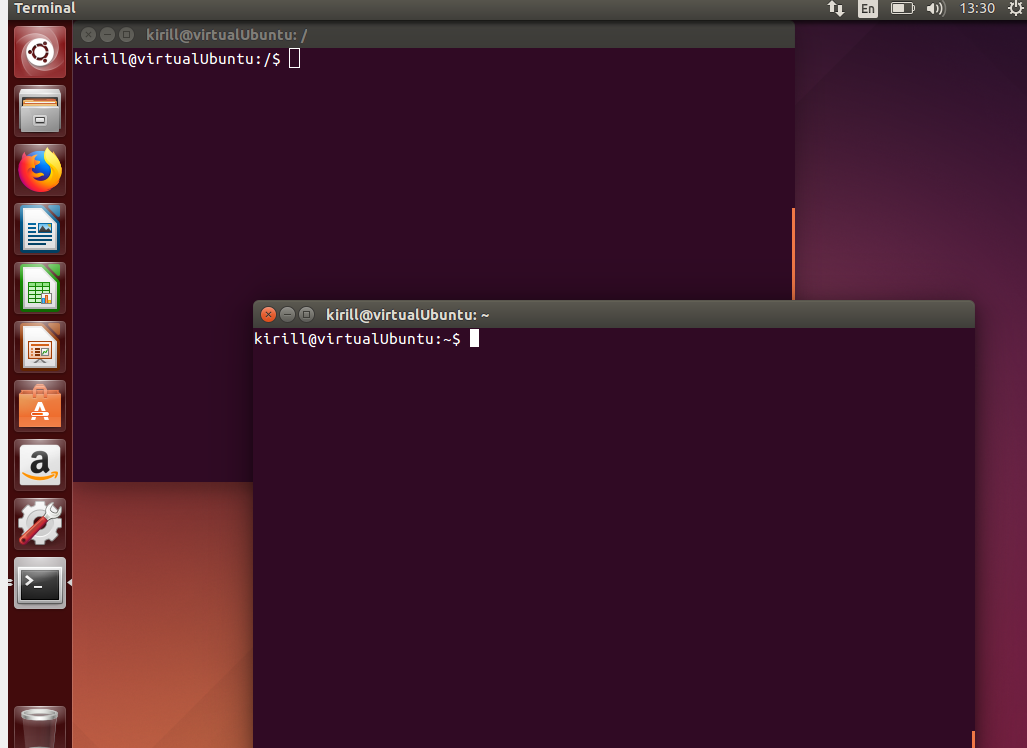
Теперь процесс находится в состоянии terminated

1. Еще раз запустим loop в фоне и убьем процесc kill -9 PID

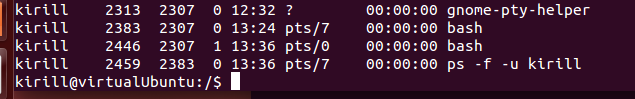


Отличие kill от terminate в том что kill всегда уничтожает процесс. А terminate дает подготовиться процессу а затем выключиться

1. Запустим еще один экземпляр оболочки bash



Процесса bash стало 2



1. Запустим несколько процессов в фоне, останавливая их, а потом запуская

