AWK

awk — это язык обработки текстовой информации. Это полноценный скриптовый язык, в нём есть конструкции условного оператора, циклов и так далее.
Синтаксис его похож на синтаксис языка C. Кроме этого, awk является предшественником языка программирования perl.

awk используется в следующем виде:
awk 'условие { действие }'
Условие необязательно.

Начнём с классической задачи: выведем Hello World.
Для этого нам понадобится действие print — вывод и условие BEGIN, которое скажет awk не ждать, когда ему передадут на стандартный ввод какие-то данные:

```
awk 'BEGIN { print "Hello, world" }'
```

awk идеален для работы со структурированными файлами. Он умеет принимать данные в виде файла либо на стандартный ввод, поэтому мы можем передать информацию через пайп от ps.

Далее вызов программы, затем в одиночных кавычках фигурные скобки, команда *print* и номер колонки, которая нам нужна.

```
ps auxf | awk '{print $1}'
```

Мы можем указать несколько колонок:

```
ps auxf | awk '{print $1 $2 $3 }'
```

На вход awk также можно подавать файл. Например, мы можем вывести всё его содержимое:

```
awk '{print }'/etc/passwd
```

Очень полезной функцией cut является возможность указать разделитель, который используется в файле; awk также позволяет это сделать:

```
awk -F ":" {print $1} /etc/passwd
```

Мы уже сталкивались с необходимостью найти определённый PID (идентификатор процесса) по имени программы в выводе *ps*. Для этого мы использовали несколько пайпов. awk позволяет нам сделать это гораздо компактнее:

```
ps auxf | awk ' /'bash'/ {print $1} '
```

Как вы видите, здесь мы использовали условие — совпадение строки.

Нумерация полей в тексте для awk начинается с единицы, но у $0 тоже есть смысл:

```
ps auxf | awk ' /'bash'/ {print $0} '
```

Такая команда выведет нам всю строку, то есть результат будет похож на результат работы grep telegram.

В awk есть **набор предопределённых внутренних переменных**.
Например, переменная NR — номер текущей строки. Используя это знание, мы, например, можем вывести третью строку из файла:

```
awk 'NR==3{print}' /etc/passwd
```

Или даже можем вывести все нечётные строки без использования циклов, а при помощи всего лишь однострочника:

```
awk 'NR % 2 != 0' /etc/passwd
```

В awk есть различные **встроенные функции**. Например, rand позволяет сгенерировать случайное значение.

Пример. Предположим, мы хотим запустить несколько задач в cron. Проблема в том, что, хотя эти процессы работают относительно недолго, они требуют довольно много ресурсов. Мы можем настроить вручную различное расписание для этих процессов, но во-первых, это лишняя работа, а во-вторых, даже если мы разнесём их на пять минут, время выполнения процесса может через некоторое время измениться, и это потребует вернуться к исправлению скрипта.

Гораздо удобнее в таком случае выставить для всех задач одинаковое расписание, например в первую минуту полудня, но при помощи sleep с рандомным числом.
Есть несколько способов получить случайное число.
Обращение к переменной:

```
echo $RANDOM
```

Далее:

```
awk 'BEGIN {srand(); print int( rand()*100) }'
```

Дальше чуть сложнее:

```
awk 'BEGIN {print rand()}'
```

Мы видим, что у нас десятичная дробь, к тому же очень маленькое число. Для того чтобы получить что-то более похожее на то, что можно использовать в качестве аргумента для команды *sleep*, домножаем число на 100 и приводим его к целому виду при помощи функции int, integer.

У нас при всех запусках возвращается одно и то же число, не очень похоже на случайные числа.

Для того чтобы числа стали меняться, и нужен *srand*: он задаёт некоторое исходное значение для генерации случайного числа функцией *rand*. Если не передавать параметр для srand, за начальное значение будет браться текущее время.

Предположим, нам необходимо посчитать количество символов в каждой строке в файле. Мы уже знаем команду *wc* с ключом -c, которая позволяет считать количество символов, и мы умеем вычитывать файл построчно при помощи while read line.
Но для этого нам нужно будет прочитать каждую строку и каждую строку передать через пайп wc.
Есть способ гораздо быстрее:

```
awk '{ print length }' /etc/passwd
```

length — одна из встроенных функций awk.

Переменная NR содержит номер строки. Мы можем пронумеровать строки в файле при помощи однострочника:

```
awk '{print NR,$0}' /etc/password
```

Предположим, у нас есть файл, в котором много пустых строк.
Здесь нам поможет **переменная NF**, которая хранит размер строки; проверим, пустая ли строка, и если нет — выведем. Так как сравнение тут строгое, строка с длиной 0 (пустая) автоматически выведена не будет. Мы можем перенаправить вывод в другой файл и таким образом создать новый файл со всей нужной информацией, но без пустых строк.

```
awk 'NF > 0' main.txt
```

Представьте, что вы случайно написали какое-то количество текста в верхнем регистре, не заметив, что включён Caps Lock, и теперь хотели бы превратить буквы в строчные. awk позволяет это сделать при помощи однострочника:

```
echo "SOME TEXT IN UPPER CASE" | awk '{print tolower($0)}' >> depts
```

Также можно перевести и в верхний регистр из нижнего:

```
echo "some text in lowe case" | awk '{print toupper($0)}'
```

Проблема может быть не во всей строке, а только в одной из колонок. Тогда мы можем применить функцию к одной колонке, а остальные вывести в таком же виде. Например:

```
awk -F ":" '{$1=toupper($1);print}' /etc/password
```