# PhamNgocHai_21002139_241008

October 8, 2024

## 1 Thực hành

### 1.1 Ex 1

Cho tệp tin "dataset.txt".

Hãy sử dụng các biểu thức chính quy kết hợp với lệnh `grep` để tìm kiếm.

[1]:
```
# 1. Tập các số tự nhiên.

!grep -E '^[0-9]$|^[1-9][0-9]*$' data/dataset.txt

##########################################################

# If accept '010' is '10' then can use
# !grep -P '^\d+$' data/dataset.txt
# Explanation
# P: Perl regex (BRE)
# ^: start of line
# \d+: one or more digits
# $: end of line

# !grep -E '\b^[0-9]+\b' data/dataset.txt
# Explanation
# E: extended regex (ERE)
# ^: start of line
# [0-9]+: one or more digits (as + is greedy, it will match as many digits as␣
 ↪possible)
# \b: word boundary

# !grep -E '^[0-9]+$' data/dataset.txt
# Explanation
# E: extended regex (ERE)
# ^: start of line
# [0-9]+: one or more digits
# $: end of line

##########################################################
```

```
# sort and filter unique for short
# !grep -oP '^\d+$' data/dataset.txt | sort -n | uniq
# Explanation
# o: only matching
# P: Perl regex (BRE)
# ^: start of line
# \d+: one or more digits
# $: end of line
# sort: sort the output
# n: numerical sort
# uniq: remove duplicates
```

```
1
5
3
2769
10
2
4
8
0
100
1000
1555
20
11000
111100
111110
100000
1000000
101000
```

[2]:
```
# 2. Tập các số tự nhiên chia hết cho 5.

!grep -E '^[05]$|^[1-9][0-9]*[05]$' data/dataset.txt
```

```
5
10
0
100
1000
1555
20
11000
111100
111110
100000
1000000
```

101000

[3]:
```
# 3. Tập các số nhị phân có độ dài 6 và chia hết cho 4.

# !echo 'Note: end = 00 or 100 -> divisible by 4'

# Method 1
!grep -E '^[01]{4}00$' data/dataset.txt
# !grep -E '^[01]{4}[0]{2}$' data/dataset.txt

# Method 2
# !grep -E '^[01]{6}$' data/dataset.txt | grep -E '00$'
# !grep -E '\b[01]{6}\b' data/dataset.txt | grep -E '00\b'
```

[4]:
```
# 4. Tập các dòng bắt đầu bằng chữ cái "T" và chứa ít nhất 2 xâu "This is an␣
 ↪exercise".

!grep -E '^T' data/dataset.txt | grep -E '.*(This is an exercise).*(This is an␣
 ↪exercise).*'
!grep -E '^T' data/dataset.txt | grep -E '.*(This is an exercise.*){2,}'

# Show the special case
!echo '\nSpecial case: T or This in the file'
!head -n 5 data/dataset.txt
```

```
This is an exerciseThis is an exercise.
T check This is an exercise about regular expression. This is an exercise in
Linux.
This is an exercise about regular expression. This is an exercise in Linux.

Special case: T or This in the file
This
abc This is an exercise This is an exercise

1
5
```

### 1.2 Ex2

Cell's Jupyter notebook doesn't support show result but the logic still work

[5]:
```
# 1. Tìm kiếm các tệp tin/thư mục trong /etc có chứa ít nhất 2 chữ cái 'a'

!ls -r /etc | grep -E '.*(a.*){2,}' > output/output1.txt
```

[6]:
```
# 2. Tìm kiếm các tệp tin/thư mục trong /etc bắt đầu bằng 'b' và không chứa 'c'

!ls -r /etc | grep -E '^b.[^c]*$' > output/output2.txt
```

[7]:
```
# 3. Tìm kiếm tên 'jiayant' đứng đầu (không phân biệt chữ hoa thường) trong tệp␣
↪"name_list.txt"

!grep -i '^jiayant' data/name_list.txt > output/out_jiayant.txt
```

[8]:
```
# 4. Liệt kê các số điện thoại có mã vùng Hà Nội "024-xxxxxxx" trong␣
↪"phone_list.txt"

!grep -E '^024-[0-9]{7}$' data/phone_list.txt > output/out_phonehanoi.txt
```

[9]:
```
# 5. Liệt kê các địa chỉ email hợp lệ trong "data_list.txt"

!grep -E '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$' data/data_list.txt␣
↪> output/mail_list.txt
```

# 2 Tóm tắt lý thuyết

`grep` command + `regular regex` + `pipeline`

Also mean:

`grep`, biểu thức chính quy (regex), và pipeline, đồng thời thêm một phần mô tả về cách sử dụng các công cụ này cùng nhau.

## 2.1 Lệnh grep

`grep` là một lệnh phổ biến trong Linux được sử dụng để tìm kiếm các chuỗi văn bản phù hợp với một mẫu trong tệp hoặc dòng đầu ra từ các lệnh khác. Cú pháp cơ bản của lệnh `grep`:

```
grep [tùy chọn] 'mẫu' [tệp tin]
```

### 2.1.1 Các tùy chọn hữu ích của grep:

- `-E`: Sử dụng biểu thức chính quy mở rộng (extended regular expression).
- `-i`: Không phân biệt chữ hoa và chữ thường khi tìm kiếm.
- `-v`: Lọc ra các dòng không khớp với mẫu tìm kiếm.
- `-r`: Tìm kiếm đệ quy trong các thư mục con.

Đọc thêm về bài tập `grep`: Grep exercises

## 2.2 Biểu thức chính quy (Regular Expressions)

Biểu thức chính quy (regex) là các mẫu ký tự được sử dụng để khớp và thao tác trên chuỗi văn bản. `grep` hỗ trợ sử dụng các biểu thức chính quy để tìm kiếm các mẫu phức tạp.

### 2.2.1 Một số ký hiệu cơ bản của regex:

- `.`: Đại diện cho bất kỳ ký tự đơn nào.
- `*`: Lặp lại ký tự đứng trước 0 hoặc nhiều lần.
- `[]`: Định nghĩa một tập hợp các ký tự (ví dụ: `[abc]` khớp với 'a', 'b', hoặc 'c').

- `^`: Khớp với đầu dòng.
- `$`: Khớp với cuối dòng.
- `\b`: Ranh giới từ (word boundary).

  Đọc thêm về tài liệu regex: Regular regex docs và các ví dụ: Regular regex examples - gen test.

## 2.3 Pipeline (|)

Pipeline (|) là ký tự được sử dụng để kết nối các lệnh với nhau. Nó cho phép đầu ra của một lệnh trở thành đầu vào của lệnh khác, giúp dễ dàng kết hợp nhiều thao tác xử lý trên dữ liệu dòng lệnh.

### 2.3.1 Ví dụ cơ bản về pipeline:

```
cat file.txt | grep 'pattern'
```

Trong ví dụ này, đầu ra từ lệnh `cat file.txt` sẽ được chuyển qua pipeline và trở thành đầu vào của lệnh `grep`, cho phép bạn tìm kiếm 'pattern' trong dữ liệu.

## 2.4 Sử dụng kết hợp grep, regex và pipeline

Dưới đây là một ví dụ minh họa việc kết hợp các công cụ trên:

```
grep -E '\b[01]{6}\b' data.txt | grep '00\b'
```

Trong lệnh này: - `grep -E '\b[01]{6}\b'` tìm kiếm các chuỗi nhị phân có độ dài đúng 6 ký tự. - Pipeline (|) chuyển đầu ra từ `grep` đầu tiên đến `grep` thứ hai. - `grep '00\b'` lọc ra các chuỗi kết thúc bằng '00'.

Tất cả các công cụ này giúp chúng ta xử lý dữ liệu phức tạp một cách hiệu quả và chính xác.

---

**More resources**: - `grep`: Grep exercises - `regular regex`: Regular regex docs and Regular regex examples - gen test - `pipeline`: How to use pipeline: Learn more about pipelines