COMPSCI 1XC3 (Winter 2026)

Lab 4-2: Regular Expressions & Text Processing

By: Nicole Sorokin

Today's Agenda

1. Lab setup
2. Searching text with `grep`
3. Basic regular expressions (BRE)
4. Extended regular expressions (ERE)
5. Text processing commands
6. TA check-off

> **Labs = Practice & Learning, not testing. Mistakes are expected and welcome!**

Create Your Lab Workspace

```
mkdir -p ~/1xc3/lab4-2
cd ~/1xc3/lab4-2
```

What is `grep`?

- `grep` = **Global Regular Expression Print**
- Used to **search text line-by-line**
- Prints only lines that match a pattern
- Works with **plain text** and **regular expressions**

`grep` does not modify files — it only searches and displays results.

1. Searching Text — `grep`

`grep` searches files for lines matching a pattern.

```
ls /usr/bin > dirlist.txt
ls /bin > dirlist2.txt
```

Basic grep Examples

```
grep zip dirlist*.txt
```

- Prints lines that contain `zip`

```
grep -l zip dirlist*.txt
```

- Prints filenames that contain a match

```
grep -n zip dirlist*.txt
```

- Prints matching lines with line numbers

Why Regular Expressions?

Regular expressions allow you to:

- Search for **patterns**, not exact words
- Match text based on **position** (start/end of line)
- Match **sets of characters**
- Handle flexible input formats (e.g. phone numbers)

They are widely used in:

- Log analysis
- Data cleaning
- Searching large text files

2. Basic Regular Expressions (BRE)

Character Matching

```
grep -h '.zip' dirlist*.txt
```

- Any character before `zip`

```
grep -h '^zip' dirlist*.txt
```

- Line that starts with `zip`

```
grep -h 'zip$' dirlist*.txt
```

- Lines that end with `zip`

Character Classes

```
grep -h '[bg]zip' dirlist*.txt
```

- Matches `bzip` or `gzip`

```
grep -h '[^bg]zip' dirlist*.txt
```

- Matches lines that are **not** `bzip` or `gzip`

3. Extended Regular Expressions (ERE)

Use `-E` to enable ERE.

```
grep -Eh '^(bz|gz|zip)' dirlist*.txt
```

- Lines starting with `bz`, `gz`, or `zip`

Matching Phone Numbers

```
echo "(555) 123-4567" | grep -E '^\(?[0-9]{3}\)? [0-9]{3}-[0-9]{4}$'
```

- Matches valid phone number format

Why This Phone Number Pattern Works

The pattern checks that:

- Area code has exactly 3 digits
- Parentheses are optional
- Phone number format is consistent
- Entire line must match (not just part)

This prevents partial or invalid matches.

BRE vs ERE (Quick Comparison)

- **BRE (Basic Regular Expressions)**
  - Default in `grep`
  - Uses simple pattern symbols like `. ^ $ [ ]`
- **ERE (Extended Regular Expressions)**
  - Enabled with `grep -E`
  - Supports grouping and alternation (`|`)

Use ERE when patterns become more complex.

Why Text Processing?

Unix tools are designed to:

- Do **one job well**
- Work together using **pipelines**

Commands like `cut`, `paste`, `diff`, and `sed` let you:

- Reshape data
- Compare files
- Transform text streams

4. Text Processing

cut & paste

```
cut -f 1 info.txt
cut -f 1 info.txt | cut -c 2-4
```

```
cut -f 1,6 info.txt > infoF16.txt
cut -f 7 info.txt > infoF7.txt
paste infoF7.txt infoF16.txt
```

`comm` vs `diff`

- `comm`
  - Compares **sorted** files
  - Shows shared vs unique lines
- `diff`
  - Shows **how files differ**
  - Used for patches and version control

Both are comparison tools, but serve different purposes.

Comparing Files

```
comm 1.txt 2.txt
diff 1.txt 2.txt
diff -u 1.txt 2.txt
```

How to Think About `sed`

- `sed` = **stream editor**
- Reads input → applies rules → outputs result
- Does not modify files unless explicitly told to

Useful for:

- Substitutions
- Formatting output
- Automated text changes

Stream Editing with sed

```
fecho "this way that way" | sed 's/that/this/'
```

```
sed '1,10s/\(^[ld-]\)\([rwx-]\{9\}\)/Type:\1 Permission:\2/' info.txt
```

TA Check-off — Task 1 (grep basics)

Goal:

Search files using `grep`.

Task (Figure the commands out and show them to me):

1. Create two directory listing files
2. Use `grep` to find lines containing `zip`
3. Show line numbers for matches
4. Show only filenames containing matches

TA Check-off — Task 2 (BRE patterns)

Goal:

Use basic regular expressions.

Task (Figure the commands out and show them to me):

1. Find lines that start with `zip`
2. Find lines that end with `zip`
3. Find lines containing `bzip` or `gzip`

TA Check-off — Task 3 (ERE patterns)

Goal:

Use extended regular expressions.

Task (Figure the commands out and show them to me):

1. Match lines starting with `bz`, `gz`, or `zip`
2. Test a valid phone number
3. Test an invalid phone number

TA Check-off — Task 4 (Text processing)

Preparing a Sample Data File

Create a sample tab-separated file:

```
cat > info.txt << EOF
l	-rwxr-xr-x	user	staff	1234	file1
d	drwxr-xr-x	user	staff	4096	file2
-	-rw-r--r--	user	staff	512	file3
EOF
```

CONTINUED: TA Check-off — Task 4 (Text processing)

Goal:

Extract and recombine fields.

Task (Figure the commands out and show them to me):

1. Extract specific fields using `cut`
2. Save outputs to new files
3. Combine them using `paste`

TA Check-off — Task 5 (diff & sed)

Preparing Files for Comparison

Create two small text files:

```
cat > 1.txt << EOF
1xc3
1xd3
1md3
1jc3
EOF
```

CONTINUED: TA Check-off — Task 5 (diff & sed)

Preparing Files for Comparison

```
cat > 2.txt << EOF
1xc3
1xd3
1dm3
EOF
```

CONTINUED: TA Check-off — Task 5 (diff & sed)

Goal:

Compare and modify text.

Task (Figure the commands out and show them to me):

1. Compare two files using `diff`
2. Generate unified diff output
3. Use `sed` to replace text in a stream