AWK

AWK OVERVIEW

- pattern scanning and processing language
- data driven not procedural
  - describe data to work with
  - tell it what to do once matching data is found
- great for dealing with files with data in columns
- we'll focus on `gawk` the GNU implementation

GAWK PROGRAM

Program has the form:

```
BEGIN { ... initialization awk commands ... }
{ ... gawk commands to run on each line of the file ... }
END { ... finalization awk commands ... }
```

- each command has the form: `pattern { action }`
- action = one or more statements enclosed in braces
- pattern can be regex
- no pattern -> action performed on all lines

FIELDS

- each line made up of fields
- field separator distinguishes fields
  - default = space
  - change value of `FS` to use other
  - or use `-F` option to change
- reference field by $# ($0 is entire line)
- `NF` = number of fields on current line
- `NR` = record number of current line

RUNNING GAWK

- `gawk [options] program [input_files]`
- `gawk -f program_file [input_files]`
- can also create a script telling it to run with gawk instead
  - `#! \bin\gawk -f`
  - needs to be executable

AWK PATTERNS

- Pattern can be regex (`/regex/`)
  - `~` used for matching regex
  - `!~` tests for not matching regex
- Pattern can also compare field or variable to value
  - `==`, `!=`, `<`, `<=`, `>`, `>=`
- `BEGIN` and `END` are special patterns
- nothing for pattern -> applies to all records
- can combine patterns with `&&` (and), `||` (or)

VARIABLES

- can hold strings / numeric values
- typically initialized in `BEGIN`
- default = initialized to empty string / 0
- standard arithmetic operators available (increment, decrement, modulo, multiply, etc.)

ASSOCIATIVE ARRAYS

- do not need to initialize
- simply refer to elements

  ex: `arr[$1] = 1`
- can test for inclusion:
  - ex: `2 in arr`
  - tests if 2 is a key in arr