AWK (CH14)

RECALL:

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

- each command has the form: `pattern { action }`
- action = one or more statements enclosed in braces

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.)

FUNCTIONS

- `length(str)` = number of characters
- `int(num)` = integer portion of num (not rounded)
- `index(str1,str2)` = starting index of `str2` in `str1` (1-based indexing)
- `split(str,arr,del)` = split `str` by `del`, place elements in `arr` (returns number of elements)
- mathematical functions are also available

FUNCTIONS (CONT.)

- `sprinft(fmt,args)` = string formatting
- `substr(str,pos,len)` = access substring
- `tolower(str)` = convert to lowercase
- `toupper(str)` = convert to uppercase

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

CONTROL STRUCTURES

if-else:

```
if (NR % 5 == 0)
  print $0
else
  next
```

for loop:

```
for (i=1; i<=NF; i++) if (i%5 == 0) print $i
```

```
for (var in arr) print var
```

OTHER PREDEFINED VARIABLES

- `OFS` = output field separator (default space)
- `ORS` = output record separator (defualt newline)
- `RS` = input record separator (default newline)