

## Line-oriented programming: awk

Three versions of awk (named after its authors — Aho, Weinberger and Kernighan): `nawk` ("new awk") is more powerful than `awk` and `gawk` (GNU `awk`) is even more powerful. I will focus on `gawk`.

### Starting gawk

`gawk options 'program' file` Or  
`gawk options -f program.file file`

When `file` is omitted, `awk` uses its standard input. It's possible to give several `-f` options.

### awk commands

Awk command has the form:

```
condition { action }
```

### awk conditions

`BEGIN` or `END` or `/reg.expr.` / Or a relation (e.g., `(NF>3)`) or a boolean expression over regular expressions and relations (e.g. `(/a.*c/ && (NF>3))`).

### awk actions

```
if (condition) statement
variable=expression
print expression
print expression redirection
next
{ statement1; ... ; statementn }
```

### awk variables

<code>NF</code>	number of fields (in the current line)
<code>NR</code>	number of records (lines) seen so far
<code>FS</code>	input field separator (see <code>-F</code> option)
<code>RS</code>	input record separator (newline)
<code>\$expression</code>	the whole input line, when <code>expression</code> equals 0
<code>\$iexpression</code>	the <i>i</i> -th field, when <code>expression</code> equals <i>i</i>

### gawk options

<code>-F fs</code>	Specify field separator (reg.expr.)
<code>-v var=val</code>	Set variable

### Example: HW2 grade confirmation script

```
#!/usr/local/gnu/bin/gawk -f
/~HW2-User: / {USER=$2; next}
/~HW2-Path: / {PATH=$2; next}
/~HW2-Total: / {GRADE=$2; next}
{next}
END {
  print "Grade: " GRADE " User: " USER " Path: " PATH >> \
    "/home/cs114/hw2/part";
  MAIL = "/usr/lib/sendmail " USER "@cornell.edu";
  print "Subject: CS114 HW2 Confirmation" | MAIL;
  print "From: nogin@cs.cornell.edu" | MAIL;
  print "To: " USER "@cornell.edu\n" | MAIL;
  print "Preliminary HW2 grade: " GRADE | MAIL;
  close MAIL
}
```

```
#!/usr/local/gnu/bin/gawk -f
{
  if (($1,$3) in saw) {
    print "Warning: duplicate " $1 " for " $3 > "/dev/stderr";
  } else {
    saw[$1,$3]="yes";
    num[$3]++;
    total[$3]+=$4;
    grades[$3,num[$3]]=$4;
    all_grades[$4]++;
  }
}

END {
  for (i in num) {
    print "Part " i ":";
    print " " num[i] " students";
    print " " average: " total[i]/num[i];
    print " " median: " grade[i,int(num[i]/2)];
    print " " stats:";
    sort = "sort -n";
    for (j in all_grades) if ((i,j) in grades) {
      printf " " j " --- " grades[i,j] " student" | sort;
      if (grades[i,j]>1) print "s" | sort; else print "" | sort;
    }
    close (sort);
  }
}
```

## Stream editor — sed

`sed -e 'script' file` or `texttised -f script_file file`  
 It's possible to give several `-e` options. When `file` is omitted, the standard input is used.

### sed substitution command

`s/reg.expr./subst.expr./options`  
 It can be prefixed by a range that has a form `addr,addr`