CSCI 330
The UNIX System

Introduction to awk

What is awk?

- created by: Aho, Weinberger, and Kernighan
- scripting language used for manipulating data and generating reports

- versions of awk
  - awk, nawk, mawk, pgawk, …

- GNU awk: gawk

What can you do with awk?

- awk operation:
  - scans a file line by line
  - splits each input line into fields
  - compares input line/fields to pattern
  - performs action(s) on matched lines
- Useful for:
  - transform data files
  - produce formatted reports
- Programming constructs:
  - format output lines
  - arithmetic and string operations
  - conditionals and loops

Basic awk invocation

- `awk 'script' file(s)`

- `awk -f scriptfile file(s)`

- common option: `-F`
  - to change field separator

Basic awk script

- consists of patterns & actions:

```
    pattern {action}
```

  - if pattern is missing, action is applied to all lines
  - if action is missing, the matched line is printed
  - must have either pattern or action

<u>Example:</u>

```
awk '/for/ { print }' testfile
```

  - prints all lines containing string "for" in testfile

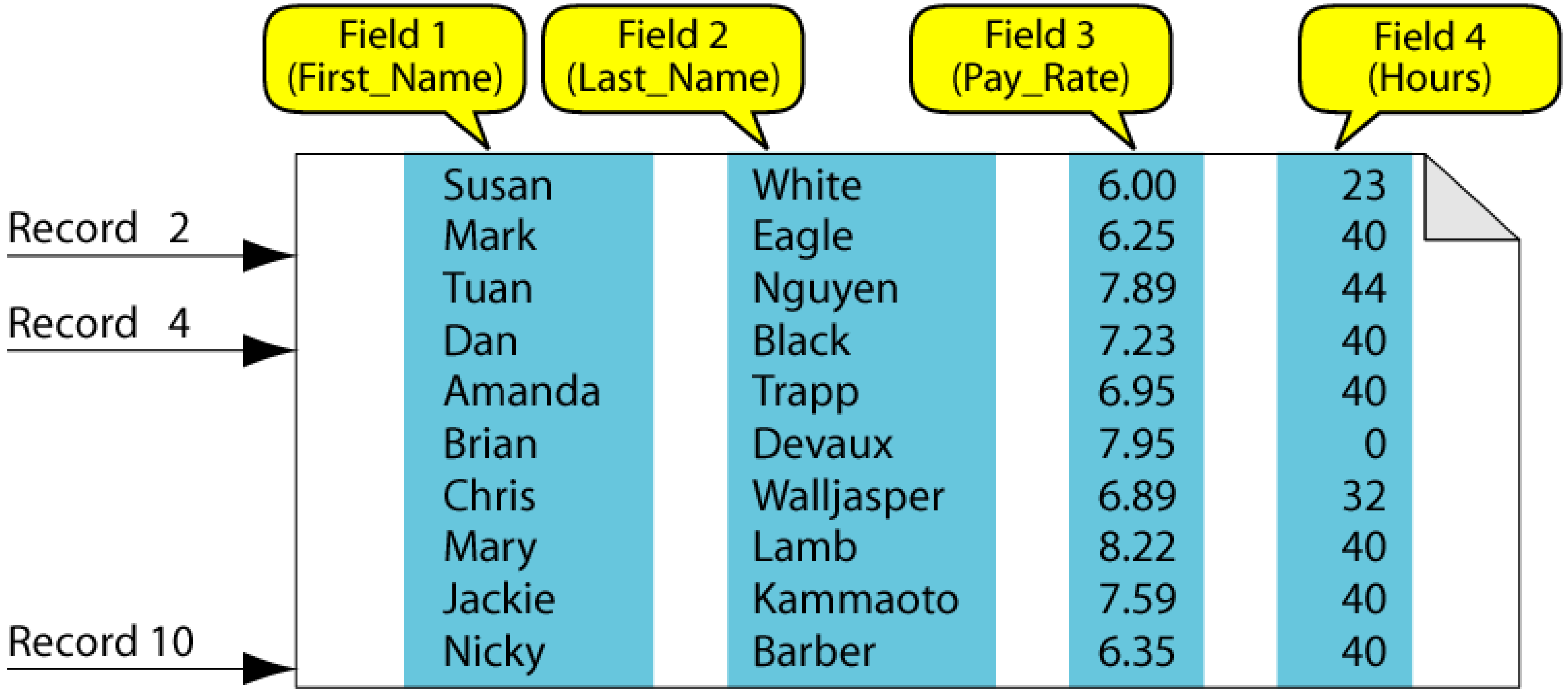
Basic Terminology: input file

- an input file is made up of records

- a <u>record</u> is the collection of fields in a line
- a <u>field</u> is a unit of data in a line

- each field is separated from the other fields by the <u>field separator</u>
  - default field separator is whitespace

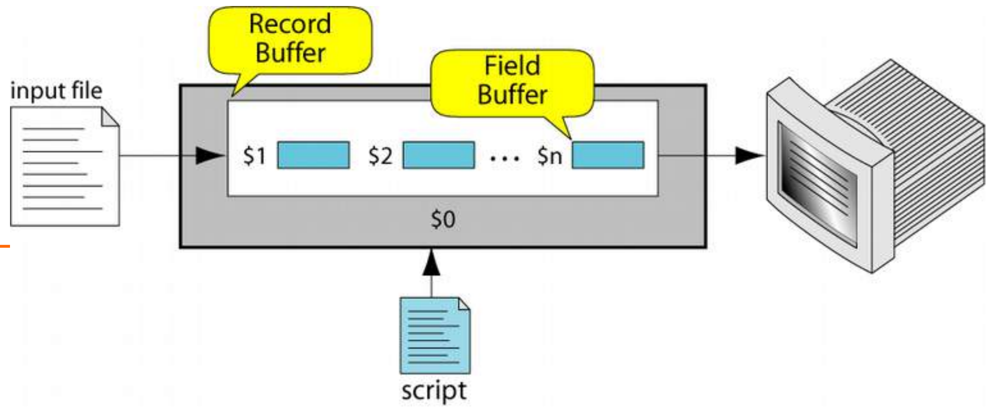
Example Input File



Susan	White	6.00	23
Mark	Eagle	6.25	40
Tuan	Nguyen	7.89	44
Dan	Black	7.23	40
Amanda	Trapp	6.95	40
Brian	Devaux	7.95	0
Chris	Walljasper	6.89	32
Mary	Lamb	8.22	40
Jackie	Kammaoto	7.59	40
Nicky	Barber	6.35	40

A file with 10 records, each with four fields

Buffer variables



- awk supports two types of buffers:
  <u>record</u> and <u>field</u>

- field buffer:
  - one for each field in the current record
  - variable names: $1, $2, …

- record buffer:
  - $0 holds the entire record

Some System Variables

NR	Number of the current record
NF	Number of fields in current record

(Note: No $ needed to read variable)

<u>also:</u>

FS	Field separator (default=whitespace)

(Can be changed during execution)

Example: Records and Fields

```
% cat emps
Tom Jones           4424    5/12/66 543354
Mary Adams          5346    11/4/63 28765
Sally Chang         1654    7/22/54 650000
Billy Black         1683    9/23/44 336500

% awk '/Tom/ { print }' emps
Tom Jones         4424    5/12/66 543354
```

Example: Records and Fields

```
% cat emps
Tom Jones       4424    5/12/66 543354
Mary Adams      5346    11/4/63 28765
Sally Chang     1654    7/22/54 650000
Billy Black     1683    9/23/44 336500

% awk '{print NR, $0}' emps
1 Tom Jones     4424    5/12/66 543354
2 Mary Adams    5346    11/4/63 28765
3 Sally Chang   1654    7/22/54 650000
4 Billy Black   1683    9/23/44 336500
```

Example: Space as Field Separator

```
% cat emps
Tom Jones        4424    5/12/66 543354
Mary Adams       5346    11/4/63 28765
Sally Chang      1654    7/22/54 650000
Billy Black      1683    9/23/44 336500

% awk '{print NR, $1, $2, $5}' emps
1 Tom Jones 543354
2 Mary Adams 28765
3 Sally Chang 650000
4 Billy Black 336500
```

Example: Colon as Field Separator

```
% cat emps2
Tom Jones:4424:5/12/66:543354
Mary Adams:5346:11/4/63:28765
Sally Chang:1654:7/22/54:650000
Billy Black:1683:9/23/44:336500

% awk -F: '/Jones/{print $1, $2}' emps2
Tom Jones 4424
```

Special Patterns

- BEGIN
  - matches before the first line of input
  - used to create header for report
- END
  - matches after the last line of input
  - used to create footer for report

example input file

```
Jan 13 25 15 115
Feb 15 32 24 22
Mar 15 24 34 228
Apr 31 52 63 420
May 16 34 29 208
Jun 31 42 75 492
Jul 24 34 67 436
Aug 15 34 47 316
Sep 13 55 37 277
Oct 29 54 68 525
Nov 20 87 82 577
Dec 17 35 61 401

Jan 21 36 64 620
Feb 26 58 80 652
Mar 24 75 70 495
Apr 21 70 74 514
```

awk script examples

- Print tables
  - select & format display of data
  - with column headings
  - with footer: number of lines processed
- Print summaries
  - line sum
  - column sum

awk example runs

- `awk '{print $1}' input`
- `awk '{print $1, $2+$3+$4, $5}' input`
- `awk '/[0-9]+/{print $1, $2+$3+$4, $5}' input`

awk example script

```
BEGIN {
        print "Mon Sales Revenue"
        count=0
}
/[0-9]+/ {
        print $1, $2+$3+$4, $5
        count++
}
END {
        print count, " records processed"
}
```

awk example script

```
BEGIN {
        print "Mon Sales Revenue"
        count=0
        sum=0
}
/[0-9]+/ {
        print $1, $2+$3+$4, $5
        count++
        sum+=$5
}
END {
        print count, " records produce: ", sum
}
```

Example: processing /etc/passwd

- field separator is “:”
- count how many users use “bash”

```
BEGIN {
        FS = ":"
        count=0
        print "list of users that use bash"
}
/bash/ {
        print $1, $5
        count++
}
END {
        print count, " users use bash"
}
```

Summary

- awk is tool to make reports
  based on data that is arranged in records/fields

- Next:
  - more patterns
  - more actions