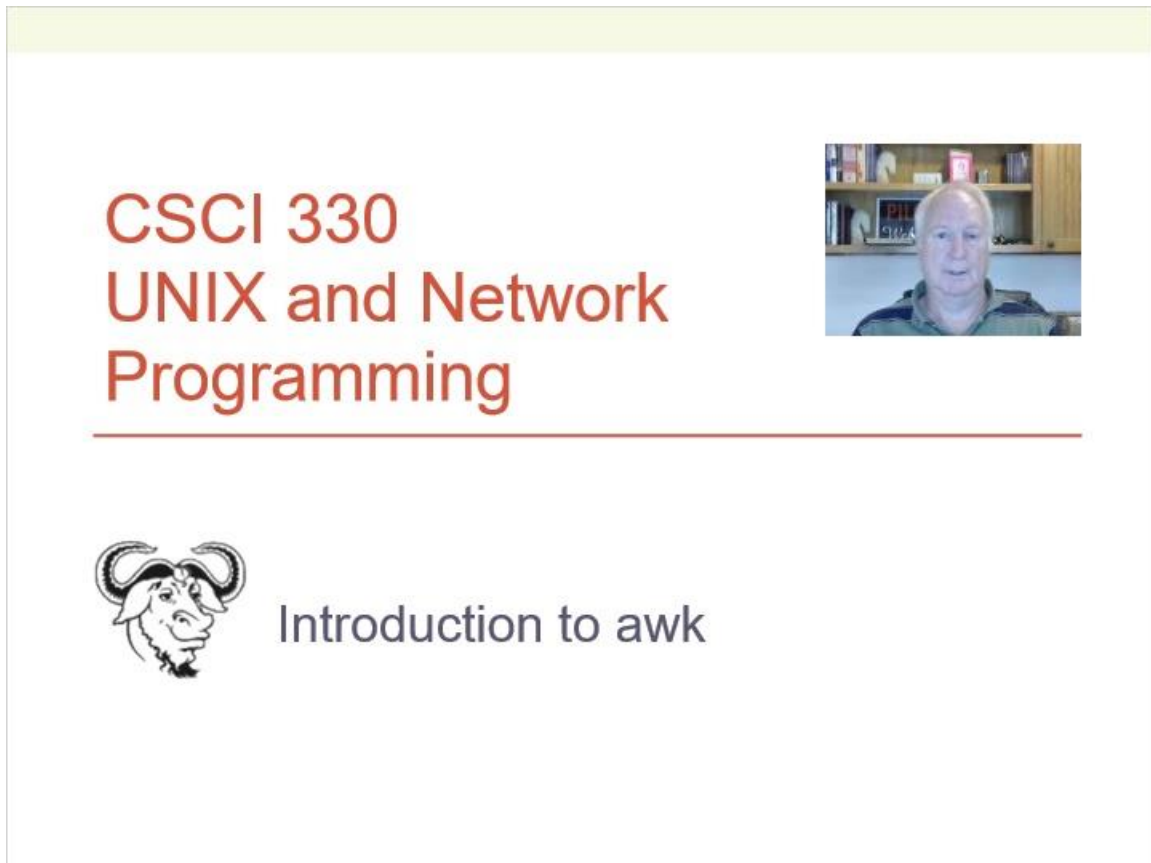


Awk Intro


1. Awk Intro

1.1 CSCI 330



A presentation slide for CSCI 330. The slide has a light yellow header bar. The main content area is white. On the left, the text "CSCI 330" is in a large, bold, red font, followed by "UNIX and Network Programming" in a slightly smaller, bold, red font. A thin red horizontal line is positioned below this text. To the right of the text is a small, square video thumbnail showing a man with white hair and a green shirt, sitting in front of a bookshelf. Below the main title, on the left, is a black and white line drawing of a ram's head. To the right of the ram's head, the text "Introduction to awk" is written in a black, sans-serif font.

CSCI 330
UNIX and Network
Programming



Introduction to awk

1.2 What is 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

1.3 What can you do with awk?

What can you do with awk?

- awk operation:
 - reads 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

1.4 Basic awk invocation

Basic awk invocation

```
awk 'script' file(s)
```

```
awk -f scriptfile file(s)
```

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

1.5 Basic awk script

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

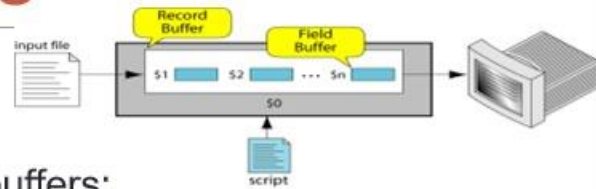
Example:

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

- prints all lines containing string “for” in testfile

1.6 awk variables

awk variables



- awk reads input line into buffers:
record and fields
- field buffer:
 - one for each field in the current record
 - variable names: \$1, \$2, ...
- record buffer:
 - \$0 holds the entire record

1.7 More awk variables

More awk variables

NR Number of the current record

NF Number of fields in current record

also:

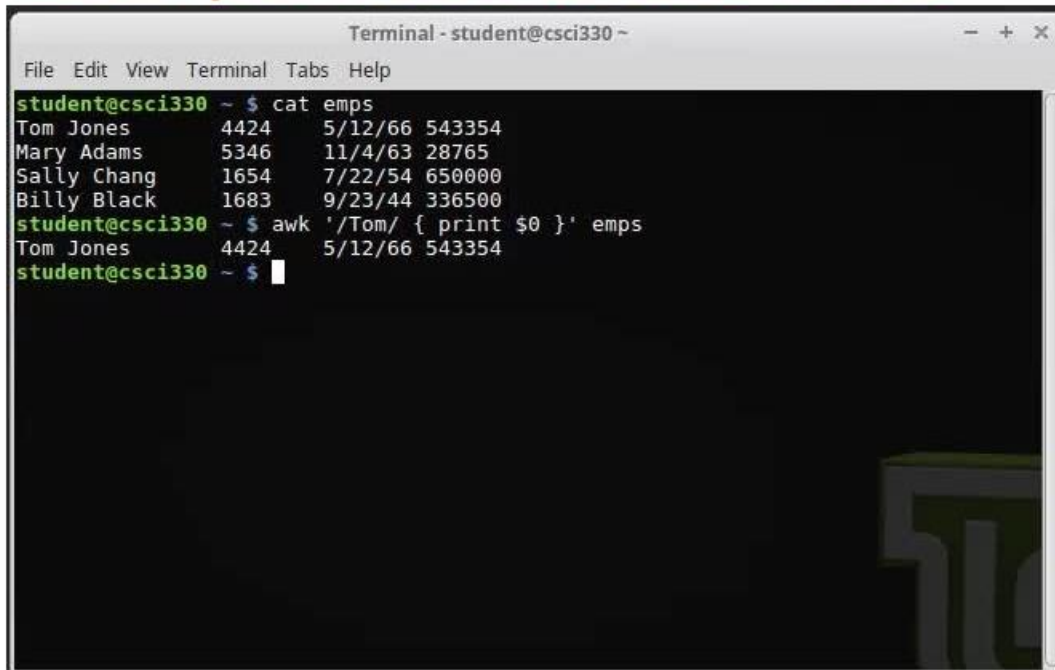
FS Field separator
(default=whitespace)

1.8 Example: Records and Fields



1.9 Example: Records and Fields

Example: Records and Fields



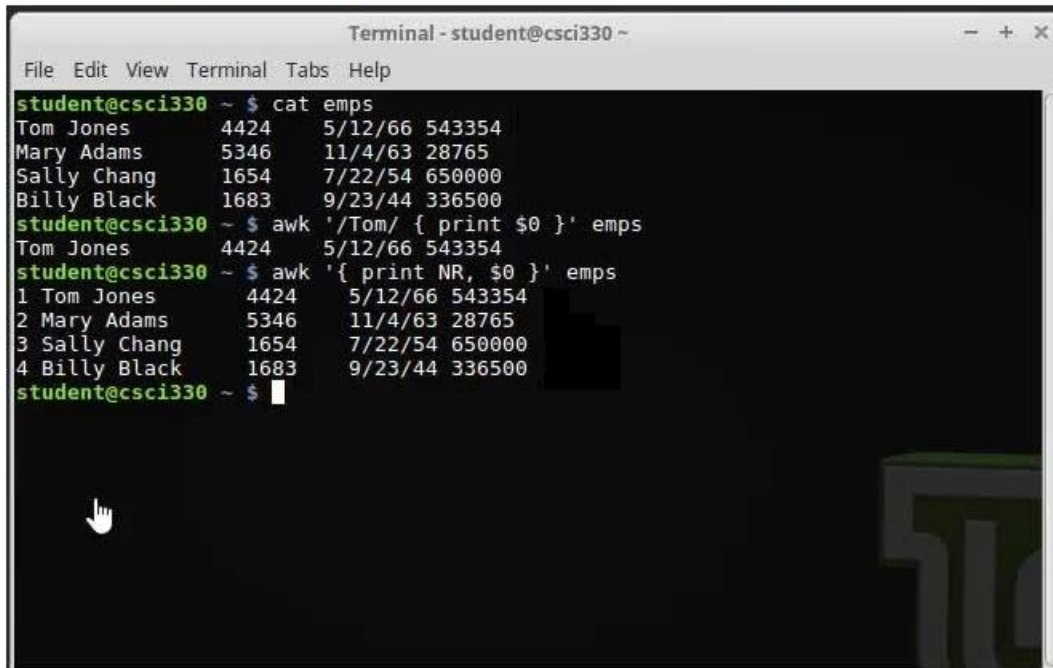
A terminal window titled "Terminal - student@csci330 ~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following commands and output:

```
student@csci330 ~ $ 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
student@csci330 ~ $ awk '/Tom/ { print $0 }' emps
Tom Jones      4424      5/12/66 543354
student@csci330 ~ $
```

The terminal output displays a list of employee records with fields: Name, ID, Date, and Salary. The second command uses awk to filter and print only the records for 'Tom Jones'.

1.10 Example: Space as Field Separator

Example: Space as Field Separator

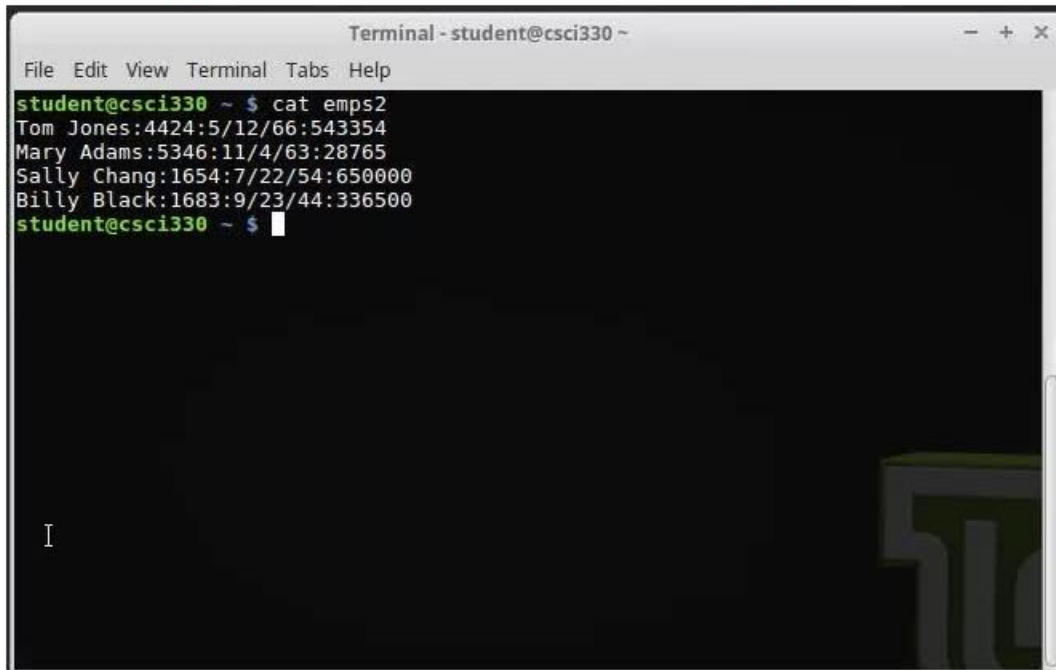


```
Terminal - student@csci330 ~
File Edit View Terminal Tabs Help

student@csci330 ~ $ 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
student@csci330 ~ $ awk '/Tom/ { print $0 }' emps
Tom Jones      4424      5/12/66 543354
student@csci330 ~ $ 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
student@csci330 ~ $
```

1.11 Example: Colon as Field Separator

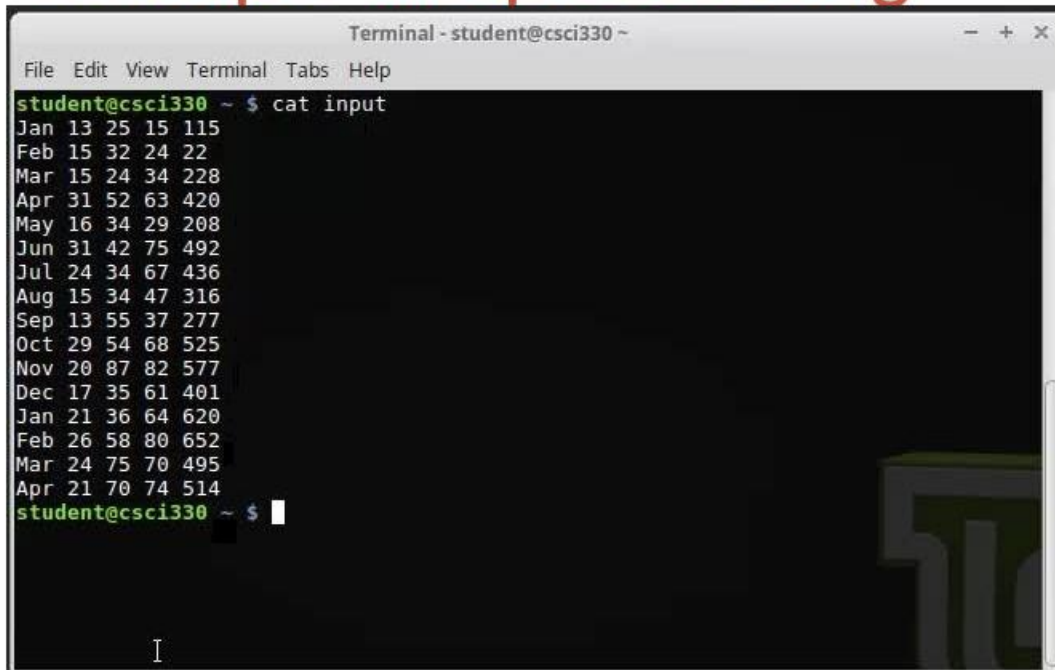
Example: Colon as Field Separator



```
Terminal - student@csci330 ~  
File Edit View Terminal Tabs Help  
student@csci330 ~ $ 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  
student@csci330 ~ $
```

1.12 example file processing

example file processing

A terminal window titled "Terminal - student@csci330 ~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is "student@csci330 ~ \$". The command "cat input" has been executed, displaying a list of months and numbers. The prompt "student@csci330 ~ \$" is shown again with a cursor. A large, faint, stylized letter 'I' is visible in the background of the terminal window.

```
student@csci330 ~ $ cat input
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
student@csci330 ~ $
```

1.13 Simple Patterns

Simple 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

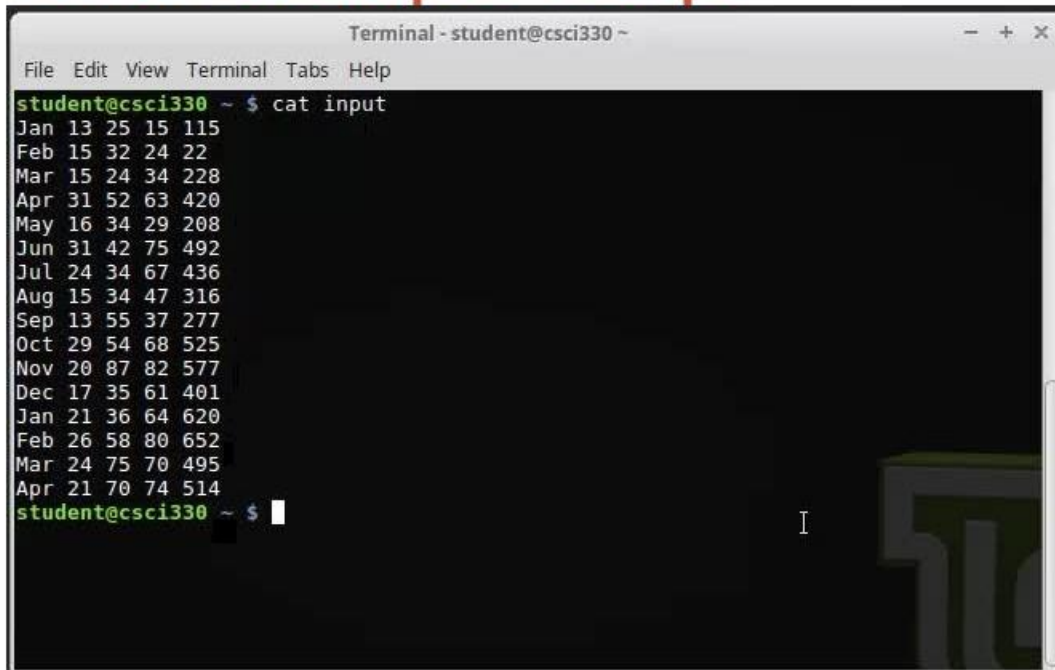
1.14 More Patterns

More Patterns

- expression patterns:
whole line vs. explicit field match
 - whole line `/regExp/`
 - field match `$2 ~ /regExp/`
- range patterns
 - specified as from and to:
 - example: `/regExp/,/regExp/`

1.15 awk example script

awk example script



A terminal window titled "Terminal - student@csci330 ~" displays the output of the command `cat input`. The output is a list of 20 lines, each containing a month, a day, and two numbers. The data is as follows:

Month	Day	Number 1	Number 2
Jan	13	25	115
Feb	15	32	22
Mar	15	24	228
Apr	31	52	420
May	16	34	208
Jun	31	42	492
Jul	24	34	436
Aug	15	34	316
Sep	13	55	277
Oct	29	54	525
Nov	20	87	577
Dec	17	35	401
Jan	21	36	620
Feb	26	58	652
Mar	24	75	495
Apr	21	70	514

The terminal prompt `student@csci330 ~ $` is visible at the bottom of the window.

1.16 awk actions

awk actions

- basic expressions
- output: `print`, `printf`
- decisions: `if`
- loops: `for`, `while`

1.17 awk Expression

awk Expression

- consists of: operands and operators
- operands:
 - numeric and string constants
 - variables
 - functions and regular expression
- operators:
 - assignment: = ++ -- += -= *= /=
 - arithmetic: + - * / % ^
 - logical: && || !
 - relational: > < >= <= == !=
 - match: ~ !~
 - string concatenation: space

awk Variables

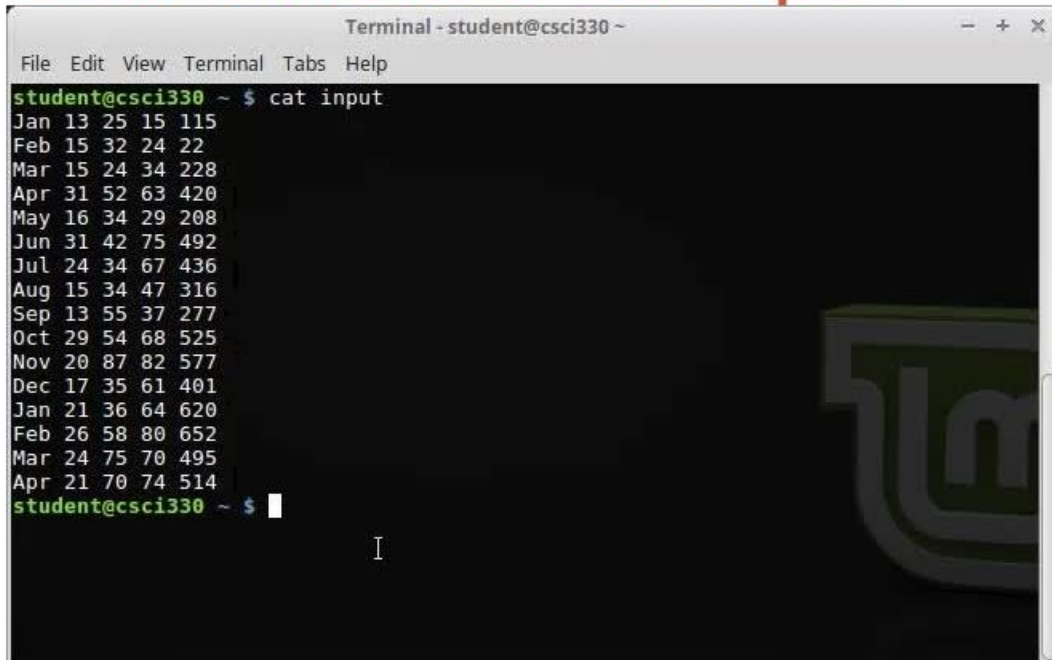
- created via assignment:

var = expression

- types: number (not limited to integer)
string, array
- variables come into existence when first used
- type of variable depends on its use
- variables are initialized to either 0 or ""

1.19 awk variables example

awk variables example



A terminal window titled "Terminal - student@csci330 ~" displays the output of the command `cat input`. The output is a list of months followed by four space-separated numbers. The months listed are Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, Jan, Feb, Mar, and Apr. The numbers for each month are: 13 25 15 115, 15 32 24 22, 15 24 34 228, 31 52 63 420, 16 34 29 208, 31 42 75 492, 24 34 67 436, 15 34 47 316, 13 55 37 277, 29 54 68 525, 20 87 82 577, 17 35 61 401, 21 36 64 620, 26 58 80 652, 24 75 70 495, and 21 70 74 514. The prompt `student@csci330 ~ $` is visible at the bottom of the terminal.

```
student@csci330 ~ $ cat input
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
student@csci330 ~ $
```

1.20 awk output: print

awk output: print

- Writes to standard output
 - Output is terminated by newline
- If called with no parameter, it will print \$0
- Printed parameters are separated by blank
- Print control characters are allowed:
 - `\n \f \a \t \b \\ ...`

1.21 print examples



1.22 printf: Formatting output

printf: Formatting output

Syntax:

```
printf(format-string, var1, var2, ...)
```

- each format specifier within “format-string” requires additional argument of matching type
 - %d, %i decimal integer
 - %c single character
 - %s string of characters
 - %f floating point number

1.23 Format specifier modifiers

Format specifier modifiers

- between “%” and letter
 - `%10s`
 - `%7d`
 - `%10.4f`
 - `%-20s`
- meaning:
 - width of field, field is printed right justified
(“-” will left justify)
 - precision: number of digits after decimal point

1.24 awk Example: list of products

awk Example: list of products

```
101:propeller:104.99
102:trailer hitch:97.95
103:sway bar:49.99
104:fishing line:0.99
105:mirror:4.99
106:cup holder:2.49
107:cooler:14.89
108:wheel:49.99
109:transom:199.00
110:pulley:9.88
111:lock:31.00
112:boat cover:120.00
113:premium fish bait:1.00
```


1.25 awk Example: output

awk Example: output

Marine Parts R Us

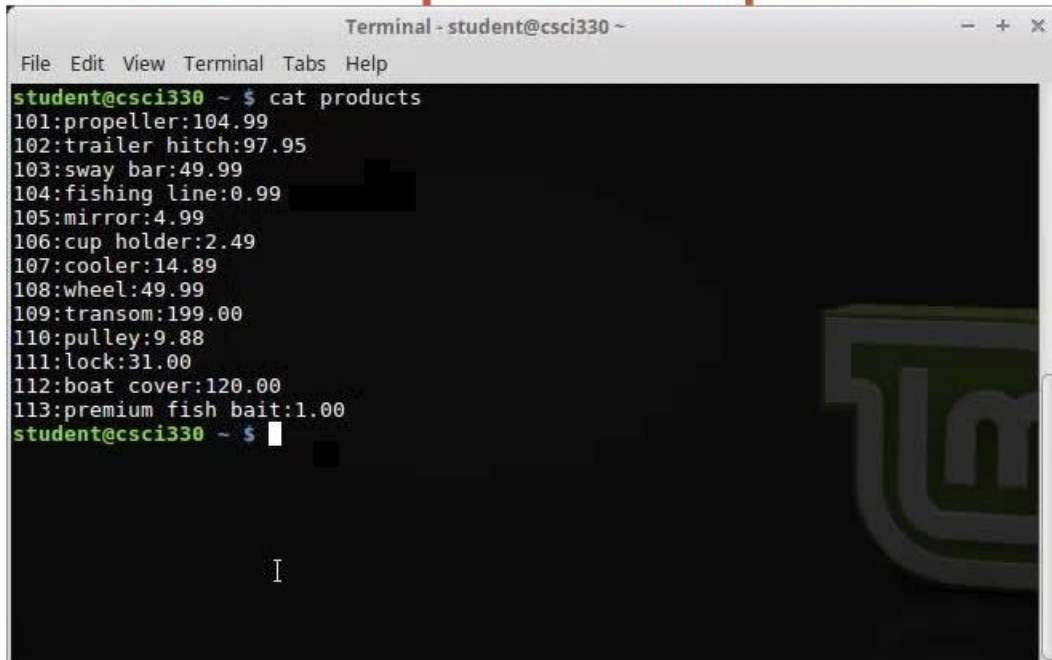
Main catalog

Part-id	name	price
101	propeller	104.99
102	trailer hitch	97.95
103	sway bar	49.99
104	fishing line	0.99
105	mirror	4.99
106	cup holder	2.49
107	cooler	14.89
108	wheel	49.99
109	transom	199.00
110	pulley	9.88
111	lock	31.00
112	boat cover	120.00
113	premium fish bait	1.00

Catalog has 13 parts

1.26 awk Example: complete

awk Example: complete

A terminal window titled "Terminal - student@csci330 ~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the command "cat products" and its output, which is a list of 13 items with their IDs and prices. The prompt "student@csci330 ~ \$" is visible at the bottom.

```
student@csci330 ~ $ cat products
101:propeller:104.99
102:trailer hitch:97.95
103:sway bar:49.99
104:fishing line:0.99
105:mirror:4.99
106:cup holder:2.49
107:cooler:14.89
108:wheel:49.99
109:transom:199.00
110:pulley:9.88
111:lock:31.00
112:boat cover:120.00
113:premium fish bait:1.00
student@csci330 ~ $
```

1.27 Summary

Summary

- awk
 - scripting language used for
 - manipulating data and
 - generating reports