Trends in Programming Language Design

16 October 2002
Trends in Programming Language Design

Overview

– The most influential languages

– Trends in language design

– Design issues in the AWK programming language
The Most Influential Programming Languages of All Time

- Assembler
  - 1950s
  - Step up from machine language
  - Available on virtually every machine
The Most Influential Programming Languages of All Time

- **Fortran**
  - 1950s
  - Created by a team led by John Backus of IBM
  - Initial focus: scientific computing
  - Influenced FI, FII, FIV, F77, F90, HPF, F95
The Most Influential Programming Languages of All Time

• Cobol
  – 1950s
  – Created by U.S. DOD
  – Grace Murray Hopper influential in initial development
  – Initial focus: business data processing
  – Influenced C68, C74, C85, PL/1
  – The world’s most popular programming language until the early 1990s
The Most Influential Programming Languages of All Time

• Lisp
  – 1950s
  – Created by John McCarthy
  – Initial focus: symbol processing
  – Influenced Scheme, Common Lisp, MacLisp, Interlisp
  – Dominant language for programming AI applications for many years
The Most Influential Programming Languages of All Time

- **Algol 60**
  - 1960
  - Algol 60 Report introduced BNF as a notation for describing the syntax of a language
  - Initial focus: general purpose programming
  - First block-structured language
  - Influenced Algol 68, Pascal, Modula, Modula 2, Oberon, Modula 3
The Most Influential Programming Languages of All Time

• Basic
  – Early 1960s
  – Created by John Kemeny and Thomaz Kurtz of Dartmouth
  – Initial focus: a simple, easy-to-use imperative language
  – Influenced dozens of dialects, most notably Visual Basic, probably the world’s most popular programming language today
The Most Influential Programming Languages of All Time

• Simula 67
  – 1967
  – Created by Ole-Johan Dahl, Bjorn Myhrhaug and Kristen Nygaard at the Norwegian Computing Centre, Olso
  – Algol 60 with classes and coroutines
  – First object-oriented programming language
  – Designed for discrete-event simulation
  – Influenced C++, Smalltalk, Java
The Most Influential Programming Languages of All Time

• C
  – 1970s
  – C was created by Dennis Ritchie at Bell Labs initially as a systems programming language for implementing UNIX
  – C++ was created by Bjarne Stroustrup at Bell Labs in the 1980s adding object orientation to C
  – Influenced ANSI C, Java
  – C/C++ has become the world’s most widely used systems programming language
The Most Influential Programming Languages of All Time

- ML
  - 1970s
  - Created by Robin Milner at University of Edinburgh
  - Initial focus: meta-language for program verification
  - One of the most widely used functional programming languages
  - Influenced Standard ML, Miranda, Haskell
The Most Influential Programming Languages of All Time

• Scripting Languages
  – Typeless languages for “glue programming”
    – awk
    – perl
    – sh
    – tkl
    – many more
Other Influential Languages

- ADA
- APL
- C#
- HTML
- Java
- PL/1
- Postscript
- Prolog
- SQL
- Visicalc
Contemporary Issues in Language Design

- Simplicity and expressiveness for productivity
- Robustness, safety and security
- Architecturally neutral and portable
- Internet savvy
- Concurrency
- Performance
- Object orientation
- Interoperability
Overview of Awk

From *The AWK Programming Language*, by Alfred V. Aho, Brian W. Kernighan and Peter J. Weinberger, Addison Wesley, 1988

“Awk is a convenient and expressive programming language that can be applied to a wide variety of common computing and data-processing tasks.”
Awk Program

• Format of an awk program
  
  pattern { action }
  pattern { action }
  ...
  pattern { action }

• Execution model
  repeatedly
  read input line
  apply patterns
  for each pattern that matches
  execute associated action
Example

Data file

<table>
<thead>
<tr>
<th>Name</th>
<th>Hours-worked</th>
<th>Hourly-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Stephen</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Susan</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Bob</td>
<td>6.5</td>
<td>11</td>
</tr>
</tbody>
</table>
How much did each person earn during their shift?

<table>
<thead>
<tr>
<th>Name</th>
<th>Hours-worked</th>
<th>Hourly-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Stephen</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Susan</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Bob</td>
<td>6.5</td>
<td>11</td>
</tr>
</tbody>
</table>

**Command line**

```bash
awk \`\$2 > 0  \{ print \$1, \$2 * \$3 \}` data
```

**Awk output**

- Bob 50
- Susan 150
- Bob 71.5
How many hours did Bob work?

<table>
<thead>
<tr>
<th>Name</th>
<th>Hours-worked</th>
<th>Hourly-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Stephen</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Susan</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Bob</td>
<td>6.5</td>
<td>11</td>
</tr>
</tbody>
</table>

**Awk program**

```
$1 ~ /Bob/ { hw += $2 }
END { print "Bob worked " hw " hours" }
```

**Awk output**

Bob worked 11.5 hours
What are everyone’s wages?

<table>
<thead>
<tr>
<th>Name</th>
<th>Hours-worked</th>
<th>Hourly-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Stephen</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Susan</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Bob</td>
<td>6.5</td>
<td>11</td>
</tr>
</tbody>
</table>

**Awk program**

```awk
{ wages[$1] += $2 * $3 }
END { for (emp in wages)
          print emp " earned $" wages[emp] }
```

**Awk output**

Stephen earned $0  
Bob earned $121.5  
Susan earned $150
What are everyone’s wages, sorted by name?

<table>
<thead>
<tr>
<th>Name</th>
<th>Hours-worked</th>
<th>Hourly-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Stephen</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Susan</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Bob</td>
<td>6.5</td>
<td>11</td>
</tr>
</tbody>
</table>

**Awk program**

```awk
{ wages[$1] += $2 * $3 }
END { for (emp in wages)
    print emp " earned "$ wages[emp] | "sort" }
```

**Awk output**

- Bob earned $121.5
- Stephen earned $0
- Susan earned $150
Awk Patterns

- BEGIN
- END
- Expression
- Regular expression
- Compound pattern
- Range pattern
Awk Actions

- expressions
- print/printf
- if (expression) statement
- if (expression) statement else statement
- while (expression) statement
- for(expression; expression; expression) statement
- for(variable in array) statement
- do statement while (expression)
- break/continue/next/exit/exit expression
- { statements }
Some useful awk “one-liners”

• Print the total number of input lines
  
  ```
  END { print NR }
  ```

• Print every line longer than 80 characters
  
  ```
  length($0) > 80
  ```

• Print the last field of every input line
  
  ```
  { print $NF }
  ```

• Print the first two fields, in opposite order, of every line
  
  ```
  { print $2, $1 }
  ```

• Print in reverse order the fields of every line
  
  ```
  { for ( i = NF; i > 0; i = i-1 ) printf("%s ", $I)
  printf("\n") }
  ```