

Professor Alfred V. Aho
Department of Computer Science
Columbia University

Trends in Programming Language Design

16 October 2002

1 Al Aho

Trends in Programming Language Design

Overview

- The most influential languages
- Trends in language design
- Design issues in the AWK programming language

2 Al Aho

The Most Influential Programming Languages of All Time

- Assembler
 - 1950s
 - Step up from machine language
 - Available on virtually every machine

3 Al Aho

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

4 Al Aho

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

5 Al Aho

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

6 Al Aho

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
 - Revised Algol 60 Report: P. Naur, J. Backus, F. Bauer, J. Green, C. Katz, J. McCarthy, A. Perlis, H. Rutishauer, K. Samelson, B. Vauquois, J. Wegstein, A. van Wijngaarden, M. Woodger

7 Al Aho

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

8 Al Aho

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, Oslo
 - Algol 60 with classes and coroutines
 - First object-oriented programming language
 - Designed for discrete-event simulation
 - Influenced C++, Smalltalk, Java

9 Al Aho

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

10 Al Aho

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

11 Al Aho

The Most Influential Programming Languages of All Time

- **Scripting Languages**
 - Typeless languages for “glue programming”
 - awk
 - perl
 - sh
 - tkl
 - many more

12 Al Aho

Other Influential Languages

- ADA
- APL
- C#
- HTML
- Java
- PL/1
- Postscript
- Prolog
- SQL
- Visicalc

13 Al Aho

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

14 Al Aho

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.”

15 Al Aho

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
```

16 Al Aho

Example

Data file

| <i>Name</i> | <i>Hours-worked</i> | <i>Hourly-rate</i> |
|-------------|---------------------|--------------------|
| Bob | 5 | 10 |
| Stephen | 0 | 8 |
| Susan | 10 | 15 |
| Bob | 6.5 | 11 |

17 Al Aho

How much did each person earn during their shift?

| <i>Name</i> | <i>Hours-worked</i> | <i>Hourly-rate</i> |
|-------------|---------------------|--------------------|
| Bob | 5 | 10 |
| Stephen | 0 | 8 |
| Susan | 10 | 15 |
| Bob | 6.5 | 11 |

Command line

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

Awk output

```
Bob 50
Susan 150
Bob 71.5
```

18 Al Aho

How many hours did Bob work?

| Name | Hours-worked | Hourly-rate |
|---------|--------------|-------------|
| Bob | 5 | 10 |
| Stephen | 0 | 8 |
| Susan | 10 | 15 |
| Bob | 6.5 | 11 |

Awk program

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

Awk output

Bob worked 11.5 hours

19 Al Aho

What are everyone's wages?

| Name | Hours-worked | Hourly-rate |
|---------|--------------|-------------|
| Bob | 5 | 10 |
| Stephen | 0 | 8 |
| Susan | 10 | 15 |
| Bob | 6.5 | 11 |

Awk program

```
{ 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

20 Al Aho

What are everyone's wages, sorted by name?

| Name | Hours-worked | Hourly-rate |
|---------|--------------|-------------|
| Bob | 5 | 10 |
| Stephen | 0 | 8 |
| Susan | 10 | 15 |
| Bob | 6.5 | 11 |

Awk program

```
{ 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

21 Al Aho

Awk Patterns

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

22 Al Aho

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 }`

23 Al Aho

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") }`

24 Al Aho