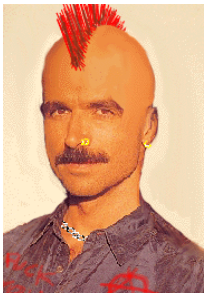
COMS W4115 Fall 2005
Project Proposal



Team Members
Joeng Kim jk2438@columbia.edu
Chris Murphy cdm6@columbia.edu
Ryan Overbeck rso2102@columbia.edu
Lauren Wilcox lgw23@columbia.edu

Introduction
This document introduces MOHAWK, a language specifically designed for processing table data in plain text files. Like AWK, MOHAWK processes text files formatted such that each row of data represents one record and white space or special characters delimit fields. MOHAWK is stream-oriented and reads input from the text files one record at a time, directing the result to standard output. For instance, a programmer can write a program to perform calculations on a set of numbers, to search through text to extract specific values. MOHAWK can transform structured data into different formatted reports.

MOHAWK can best be described as AWK--++. That is, it starts with some of the basic functionality of AWK, leaves out some of the features, and then adds some enhancements. For instance, MOHAWK supports basic AWK features like reading a data set from a text file, basic mathematical operators on the fields, printing to standard out, conditionals, loops, and simple regular expressions for pattern matching.

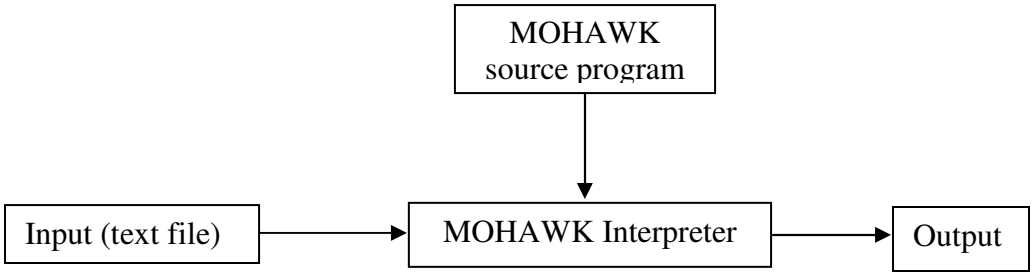
On top of this, MOHAWK adds a few new features. MOHAWK gives the programmer the ability to assign meaningful variable names to the field identifiers (e.g., "price" instead of "$2") and introduces other improvements to the syntax to make it easier to read and understand.

The name MOHAWK originally came from the team members' surnames, with an intentional nod to AWK, its predecessor. Although Kernighan noted that "naming a language after its authors ... shows a certain poverty of imagination", we felt that the name (and the process of naming it) should stay true to its roots.

Feature Set
MOHAWK will support the following AWK features:
- Read a data set from a text file, parsing records and fields
- Add, multiply, subtract, divide and reorder fields
- Declare variables for storing global and local data
- Use global constants to get metadata like the number of fields
- Print to stdout
- Use constructs such as conditionals and loops
- Use arrays of data
- Perform pattern matching using simple regular expressions

In addition, MOHAWK will add the following new features:
- Labeling fields with meaningful identifiers
- Global constants to provide an automatic rowcount

Architecture
Like AWK, MOHAWK is an interpreted language, as opposed to a compiled language like C. The critical parts of the MOHAWK interpreter will be the lexer, parser, and tree walker, the last of which will actually execute the MOHAWK code.



Syntax
MOHAWK will use a syntax that is very similar to that found in AWK, with a few adjustments. Because most modern programmers are familiar with C and Java programming syntax, MOHAWK will take advantage of that as much as possible.

Following are some of the main features of the MOHAWK syntax:

- The first part of the program assigns variable names to the fields in the data source
- The optional "BEGIN" keyword specifies a routine that can be executed before any input is read
- The "MAIN" keyword surrounds the main body of code, which loops for each record, using a pattern/action procedure that is applied to each input record
- The optional "END" keyword indicates the code that runs after finishing all of the records
- Like AWK, there are no data types: the same variable can represent a character, string, integer, etc.
- A semi-colon indicates the end of a line
- Java-like comments
- Functions use parentheses around arguments

Sample Program #1: Employee payment calculation

Datafile:

```
Bob  40 8.75
mary 43 9.65
```

Program:

```
BEGIN \
{
// assign names to fields
name = $1;
hours = $2;
rate = $3;

// declare other variables
sum = 0;
}
MAIN
{
   overtime = 0;
   if (hours > 40)
   {
```

```
      overtime = hours - 40;
      hours = 40;
   }

   // employees get time-and-a-half for overtime
   total = (hours * rate) + (overtime * rate * 1.5);
   println(name + ": $" + total);
   sum = sum + total;
}
END
{
   println("Total wages: $" + total);
}
```

Output:

```
Bob: $350
mary: $425.425
Total wages: $779.425
```

Sample Program #2: Grade calculator

Datafile:

```
Amy   95 93 92 97 83
bill  78 93 91 80 88
cindy 94 86 92 91 88
```

Program:

```
// assign names to fields
name = $1;
grade[] = $2, $3, $4, $5, $6;

// declare other variables
total = 0;

MAIN
{
   sum = 0;
   // sum up the grades
   for (i = 0; i < grade.size – 1; i++)
   {
      sum = sum + grade[i];
   }

   // find the average... NF = number of fields
   avg = sum/(NF – 1);
   println(name + ": " + avg);
   total = total + avg;
}
END
{
   // find the class mean... NR = number of records
   mean = total/NR;
   println("Class mean: " + mean);
}
```

Output:

```
Amy: 92
bill: 86
cindy: 90.2
Class mean: 89.4
```

Project Plan

Following are the planned major milestone dates in the development of MOHAWK:

- September 27: Deliver initial proposal
- October 4: Agree on architecture, including technology decisions (development platform, source control, bug tracking, etc.) and coding conventions
- October 6: Agree on MOHAWK syntax; Start writing language reference manual
- October 10: Start development of lexer/parser with ANTLR
- October 20: Deliver language reference manual
- October 31: Start development of tree walker and back-end; Start creation of test plan
- November 7: Start testing of lexer/parser
- November 14: Start testing of tree walker and back-end
- November 28: Complete development of lexer/parser; Start writing final report
- December 5: Complete development of tree walker and back-end
- December 12: Complete all testing
- December 20: Deliver final report