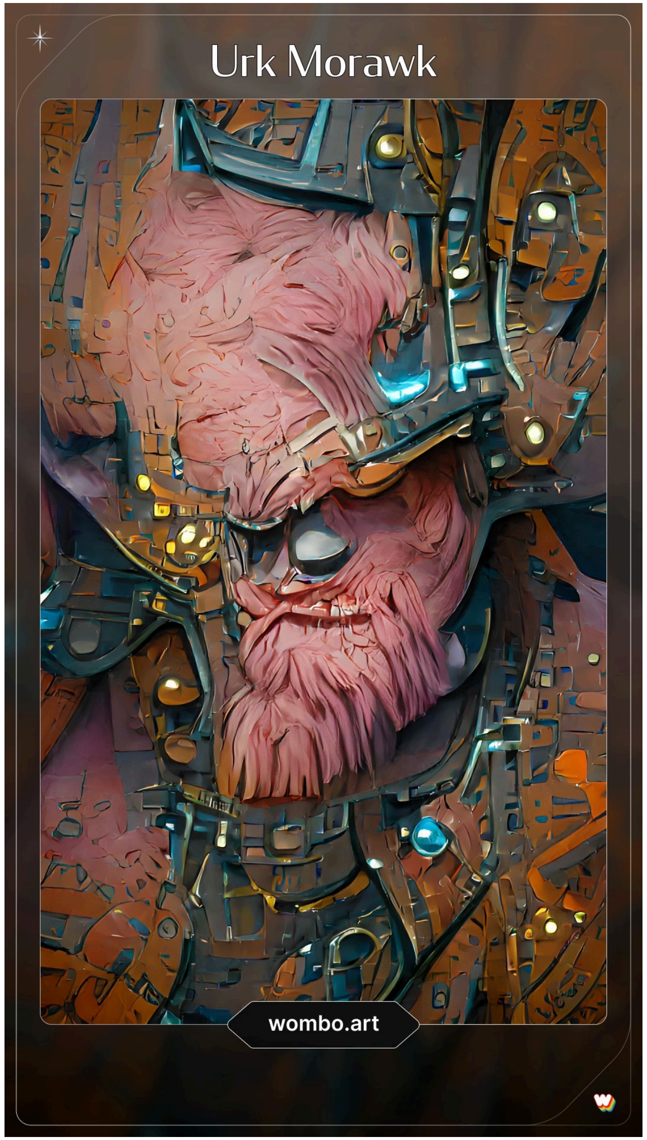
THE UNIVERSITY OF WESTERN AUSTRALIA

Morawk

Lecture 15

Michael J Wise

Urk … Morawk



General For Loop

- Awk also has a general for loop as we often need to iterate over other things that just arrays
- Format is the same as C's.

`for(`*<initial expr> ; <continuation test> ; <update expr>)*

<statement or statement block>

- If required, the *<initial expression>* sets up the loop, e.g. by initializing a loop variable.
- Immediately after initialization and after each iteration the *<continuation test>* is evaluated. If FALSE, the loop terminates; otherwise computation continues to the first/next iteration.

General For Loop

- At the end of each iteration, just before the <*continuation test*>, the <*update expression*> is evaluated. It generally updates some loop variable.
- Any of the three parts may be absent, but the structure including the semicolons must be present

```
BEGIN{ sum = 0}
{for(i=1; i<=NF; i++)
    sum += $i
}
END{printf(”Total across the file: %d\n”, sum)}
```

Built-in Functions

- Awk has a number of very handy building functions, including:

`length(`*<string>*`)` *#return the length of the string*

`substr(s, m, n)`

- substring of `s`, starting at `m`, with length `n` (or till end, if `n` absent)

`sprintf(fmt, expr, …)`

- Like `printf`, but returns a string rather than printing. Only way to concatenate strings within Awk

Built-in Functions

`sub(r, t, s)` # Substitute t for first regex r in s

`gsub(r, t, s)` # Substitute t for all regex r in s

- E.g. `gsub(″  *″, ″ ″, $1)` # multiple spaces
- These two commands allow you to do in Awk much of what you can do in Sed

`system(cmd)`

- Execute Unix command `cmd`
- Returns the exit status
- Tactic can be to create a Unix command string which results in a file, an then use `getline` to reimport the new data back into Awk, e.g. `sort`

Demo

- `my_cut.awk` is a small Awk script that reports a named column from a text file. Note the use of command line option `–v` to import value

```
# report a column specified by command line variable
# e.g., -v COL=3 (default first column)
BEGIN{
  if(COL == "")
    COL = 1
}

NF >= COL {print $COL}
```

Example:

```
% gawk -v COL=3 -f my_cut.awk  jab.txt
```

Demo

- Extracting a single column using Awk is straightforward (and more flexible than Cut).
- Write a Gawk script, `excise_col.awk`, which excises a column and reports whatever is left
- Once again, the identity of the column is imported by the command line argument, `-v COL = …`
- Hint: Needs two `for` loops

User Defined Functions

The format for Awk functions is:

```
function <name>(<parameter list>)
{
<statements>
}
```

User Defined Functions

- For example:

```
function max(a, b)
    {
    if(a>=b)
       return(a)
     return(b)
    }

{print max($1, $2)}
```

About User Defined Functions

- The open-bracket for the parameter list must immediately follow the name of the function (no spaces between).
- If there are no parameters, the brackets must still be present.
- The parameter list is a comma-separated list of names.
  - *Variables in the parameter list are local to the function (i.e. distinct from other variables found in the awk-script).*
  - *All other variables in a function definition are global!*

About User Defined Functions

- Scalar values, numbers, strings, etc, are copied to function parameters (i.e. call-by-value).
- Arrays are passed to function parameters as call-by-reference, i.e. a reference to the original array is passed, so alterations to array elements apply to the original array.
- Recursion is permitted.

`return` <*expression*>

- is used to return a value to the function caller (otherwise the return value is undefined)

Why Have User Defined Functions?

- Literate programming (again)
  - *Think of the poor folk who will have to read your code and maintain it*
- Problem decomposition
- Saves retyping identical text, e.g. multiple calls to max function

Demo

- Awk does not have max function, i.e. given an array, return value of the largest element. Write a function to compute `max_array`.