Running Head: PIPE-AND-FILTER

Programmatic Plumbing: Pipe-and-Filter

Jered McClure

Walden UniversityProgrammatic Plumbing: Pipe-and-Filter

In the beginning was the words, and the words were punched into paper, and the paper turned into cards, which eventually became programs. The only way these programs could be linked together was via means of programmatically storing data and then referencing that stored data in the next application chain. However in 1972 a man by the name of Doug Mcllroy developed the idea of pipelining data (Bergen, NA).

 “Pipes were first suggested by M. Doug Mcllroy when he was a department head in the Computing Science Research Center at Bell Labs” (The Linux Information Project, 2004). After much deliberation, and reviewing of Mcllroy’s design, Ken Thompson rewrote portions of the Unix OS to include pipelining data, this was in 1973 (The Linux Information Project, 2004). That being said, their version was a linear pipeline system, specific to writing programmatically on a single command line.

 A pipeline, or Pipe-and-Filter as it is commonly known, links the output of one program to the input of another program. “This direct connection between programs allows them to operate simultaneously and permits data to be transferred between them continuously rather than having to pass it through temporary text files or through the display screen and having to wait for one program to be completed before the next program begins” (The Linux Information Project, 2004). Microsoft has a similar version to the one created by Mcllroy, for the UNIX platform, used in the Windows OS. However, the Microsoft version does not immediately pass the stdout (standard out) to the next stdin (standard in) until the running program has completed (The Linux Information Project, 2004).

 “The general syntax for pipes is: command\_1 | command\_2 |[| command\_3 . . .]” (The Linux Information Project, 2004). The plus side of this is that Filters can be isolated and, as such, protected from the workings of the entire pipeline. Also, filters can, themselves, be made up of further pipelines, thereby reducing redundant coding where required. The downside is that if filters require all data before running, or the pipes only accept one data type at a time, than the process may slow down or come to a complete halt (Bergen, NA) (March, 2003). This can be seen when running a search command. The search must complete before the next filter in the pipeline can be initiated.

 Before pipelines, the best way people had of linking programs together was by creating an output file which could then be processed as part of the next “input” program. This was by no means a pipeline as it is today. Due to the nature of programmatic solutions at the time, these output files had to be part of the input program themselves. This was mostly due to the fact computational resources were stored on punch cards or another low density medium.

 Overall, the Pipe-and-Filter, or pipeline, method of programmatically creating batch files has come a long way since the days of punch cards (Kloth, 2003). The pipeline is one of the foremost syntaxes when it comes to batch files as well (Woude, 2011). Previously, the only way to come anywhere near the flexibility of a pipeline was by creating a program and referencing it in another program. While the pipeline may not be a series of tubes, it does pipe data from one programmatic sequence to another in the fastest possible manner (Steenson, 2009) (The Linux Information Project, 2004).

Reference

Bergen, P. V. (NA). *Pipe-And-Filter.* Retrieved February 10, 2011, from Garfixia Software Architectures: <http://www.dossier-andreas.net/software_architecture/pipe_and_filter.html>

Kloth, R. D. (2003, April 15). *Cardpunch :punch a punched card.* Retrieved February 11, 2012, from KLOTH.NET: <http://www.kloth.net/services/cardpunch.php>

March, D. L. (2003, April 11). *Pipe-And-Filter Style.* Retrieved February 10, 2012, from IT533 - Software Architectures - Class 3: <http://www4.desales.edu/~dlm1/it533/class03/pipe.html>

Steenson, M. W. (2009). *A Series of Tubes.* Retrieved February 10, 2012, from Ignite: <http://igniteshow.com/videos/series-tubes>

The Linux Information Project. (2004, April 29). *Pipes: A Brief Introduction.* Retrieved February 10, 2012, from The Linux Information Project: <http://www.linfo.org/pipe.html>

Woude, R. v. (2011, March 4). *Redirection.* Retrieved February 10, 2012, from Rob van der Woude's Scripting Pages: <http://www.robvanderwoude.com/redirection.php>