Grep: A Grammar

by Loss Pequeño Glazier

The working through of any real process will contain a sequential logic according to its own particular, essential dynamic. The character of that dynamic, which it acquires only in that exact and self-same process, becomes its own definition.

Johanna Drucker

Larger productions, such as poems, are like completed machines. Any individual sentence might be a piston. It will not get you down the road by itself, but you could not move the vehicle without it.

Ron Silliman

If you are writing ascii in the c shell along the forbidding but captivating coast of unix, the grep command will be an essential tool at your side. Grep is a powerful command though one that is used mostly for file maintenance. Of interest here are other uses of grep, especially its use as a writing *method*; a concept that would probably leave a lot of unix programmers scratching their cpus. Such a vision of grep makes particular sense if we can begin to move from the concept of writing as the production of something static (a codex or file) to a

Loss Pequeño Glazier is the director of the Electronic Poetry Center. He has written a forthcoming book on digital poetics and has numerous published poems, essays and kinetic works.

more dynamic or fluid concept of writing as the *action* of production (process[1]). That is, to a viewpoint where it's the procedure or algorithm that counts, the output being simply a by-product of that activity. This is not an easy undertaking since so many of our assumptions about textuality rely on a traditional "fixed" scheme. What exactly is a grep and how can it have relevance to the production of text?

Grep is one of numerous similar programs, ed (a line editor) and sed (a "stream of text" editor), among them, along with the siblings of grep, `fgrep` and `egrep`. Such programs handle text in terms of input and output, and perform their activity with given modifiable parameters. At its lowest level, the grep is a "find" command. "Use grep to search for text in files, or in the standard input, then display each line matched on the standard output." (Topham 70) It is also sometimes described in terms of patterns: "Grep searches files for a pattern and prints all lines that contain that pattern" (Newnes 312). Commands, though often considered counterintuitive, are quite simple. For example, to search for lines that contain the word "halibut" in a file called "fish", you would enter:

```
grep halibut fish
```

at your command line. If the source file "fish" contained only the following 3 lines:

```
smoked salmon sings
sad halibut sags
grand carpe diem
```

Then the output of grep halibut fish would be:

```
sad halibut sags
```

That is, grep will use the pattern word as a filter to return lines (in this case only one) that contain that word.

One could argue that this command is solely geared towards locating. However, it has greater possibilities. First, grep is a rigorous text processing procedure. In certain senses, this would reinforce the popular impression of UNIX as an unforgiving and inhospitable system (especially for infrequent users). On the other hand, such rigor is interesting in the post-DOS era where computers frequently crash for no apparent reason.[2] (These crashes can result from pile-ups among multitudes of invisible programs, i.e., for reasons the user can't even see.) Such a rigor bespeaks a direct relation to the electronic text, a fearful symmetry, indeed. Yet one with productive qualities as strict rules, definition,

consummately effective flags, and undeviating execution of procedure, result in starkly literal (and material) textual output.

Such materiality is evident in concrete conceptions of language: "literal strings," "strings," "regular expressions," and "compound expressions" are among the way language is viewed in the world of grep. (Grep patterns are constructed from "regular expressions", that is, "ordinary" non-binary characters such as digits, letters, and various characters used as operators, the fundamentals "materials" of writing.) Materiality is also foregrounded by the intensified sense of grammar employed to evoke grep activities. Such a grammar, once invoked, sets into motion a determined process of textual production. One thus writes instructions for the *process*, and the output text is simply a by-product. As Johanna Drucker has written on other process work, "It is occurring then carries through whatever growth is appropriate to the logic of its own development.... That becomes the logic of its own development, the way grammar becomes an absolute fact of language: because the words are such powerful objects they command relation — or is it more simple, even, they are the units and any sequence of units becomes a structure" (264-5). The grammar of grep is one where expressions are quite literal. For example,

```
grep -i halibut fish
```

would retrieve both "halibut" and "Halibut".

```
grep -l hal* fish/*
```

would retrieve anything beginning with the letters "hal" in any file in the directory "fish" and only output filenames of occurrences, not actual lines of text. The shorthand for some of these possibilities is expressed by

```
grep [-bchilnsv] pattern file(s)
```

an arcane looking proposal indeed yet one that lays out operational directions for these rigorous procedures; it is one that also seems to invoke Ron Silliman's assertion that "Structure is metaphor, content permission, syntax force" (57).

Such rigor emphasizes the line as the basic unit of text. This could be called a "phrasiform as opposed to word form" (264) conception, as Johanna Drucker has written on another procedural investigation. It is interesting to work in texts where the basic unit is the line; some would argue this has special relevance to poetry. For UNIX, there are historical reasons for this; namely that grep is descended

from ed, an editor that displays text on a line by a line basis rather than on a complete screen. Such a concept evokes the use of teletypes instead of terminals and dates from a time when a key concern was keeping the amount of data transmitted to a minimum so that communication lines would stay open and costs would be kept down. The mechanics of the teletype medium, like the medium of International Code of Signals explored by Hannah Weiner in works such as *Code Poems*, create a potential objectification of language, thus providing access to its materiality. Such explorations provide, in Charles Bernstein's words, a "radical reaffirmation of a commitment to writing as a specific kind of object making, an investigation rather than an aestheticization" (286). The mechanics of grep provide a medium with consistent properties; this fact is important for the production of writing that emphasizes its material qualities.

The secondly reason grep has greater possibilities than being merely a search tool is that it treats the text *as a file* and operates on the text at a non-represented (or more material) level. One must image a dual world: the visible or tangible level (surface) presents interpreted or represented objects (pages) on the level of interpretation (the Web browser or through Windows). "Location" here can be points within individual documents or points within documents on different servers. The second level is the non-displayed. On this more chthonic level, objects are uninterpreted (files) and exist as the material surface at the location where writing "takes place" (through the editor). One might think of this as an intangible or invisible place but the fact is that it is *not* invisible; it is as present as the representations of browsers. In fact it might be *more* tangible since these are files composed entirely of "real" text ("regular expressions") with no control characters.

For one engaged in the production of representation, grep is a simple tool and would not generally be thought of as a text production engine. Similarly, someone painting portraits of judges for display in a courthouse might not think of a can with a hole in it as a painting utensil (as Jackson Pollack would). Like the paint can and many other UNIX utilities, the grep command is a solid conduit for text with strictly defined rules and properties, rigorous in its execution of procedure. Like the hole in Pollack's paint can, a grep is an opening into the world of the materiality of words constituting the electronic text file.

What possibilities might there be for grep and poetry? Let us consider a few examples.

Grep Experiments

For instance one could just as easily take the TURK POEM *and make an* AEROPLANE *and vice versa the possibility of such transformations lies less on the nominal level and more in the material consistency even in this particular case it is quite easy one folds the turk poem according to the pattern for making an aeroplane and ends up with an admittedly highly simple but airworthy aeroplane.*

—Oskar Pastior

Most importantly, a grep is a procedure. Like the procedures of Oulipo, Cage, Mac Low, or procedures such as those suggested in Bernstein's poetry "Experiments" list, a grep is a formal method or program for parsing or altering the machine-readable text according to a fairly basic algorithm. Grep tells the machine: "Check the first line of text, if that line of text contains x, output that line; if there is following line of text, check that one and repeat the procedure; otherwise, stop". In fact, the simplicity of this procedure makes it seem merely utilitarian. But its ability to unadornedly incise text makes it fundament to any investigation of text-altering programs. Following are three examples of how procedures can be conceived for grep. (See http://wings.buffalo.edu/epc/authors/glazier/greps/ for some versions of these works.)

A Grep for Jackson Mac Low

A grep of "Methods for Reading and Performing Asymmetries" was first published in part in *Crayon* 1 (1997). The source text for this work consists of Jackson Mac Low's "Methods for Reading and Performing Asymmetries" as it appears in *Representative Works: 1938-1985*. (Lines were kept consistent with the original to allow comparison with the published text.[3]) The UNIX grep command was used to produce blocks of text from pattern words drawn from the title, "Methods for Reading and Performing Asymmetries", with the omission of "for" and "and". This title is also generated as the first line of each of the output sections since this title begins each section of the source text. Date of generation was 5 August 1997. Use of the grep was a way to examine a procedural work through a procedure.

"The Fishtail" & "Grep on 'Y'"

These two works are both derived from writings of Hawthorne and

Melville, with an emphasis on Melville's letters to Hawthorne[4]. Parts of the multi-section poem, "The Fishtail: Arrowhead, or 'The Whale'", were generated through the grep procedure: (1) In section "M", "Shakespeare" was used as a pattern word to generate the indented text from the source text "Mosses from an Old Manse." (2) The epigraph to Section h is a grep of the pattern word "church" on the source text *Moby Dick*. (The output was reformatted for paragraph style). (3) A grep of the serendipitously selected pattern words "manse," "book," "house," and "mosses" on the source text "Mosses from an Old Manse" generated the section "Manse/Mosses". (This section also treats the source html file as a literal text, allowing html encoding to result from the grep.) (4) Finally, "from Melville's Letters to Hawthorne" is a grep on the twelve surviving letters from Melville to Hawthorne[5] as available at "The Life and Works of Herman Melville" (http://www.melville.org/corresp.htm). Pattern words for this grep, selected serendipitously, were "gable", "book", "mount", "gray", "dard", "shore", "house", and "pit". Except for the reformatting into paragraph format noted above, no post-grep alterations were performed on the output. Using the grep procedure was a way to "collaborate" with these texts influenced by literary exchange.

"Ego non baptiso te in nomine: A grep on Y" was first published in *Salt* 10 (1997) as part of "[Mayapán] : a Poetics of the Link". This is a very simple grep, using a single pattern word consisting of one character, "Y". The source text consists of the twelve surviving letters from Melville to Hawthorne. "Y" was selected as a pattern word due to its vocalic kinship with the personal pronoun "I", since the "I" is germane to these letters. (The letter "I" could not be used due to the inordinate amount of output it would have produced.) No post-execution editing was performed.

"Clear Eyd Fox Quickn Brown Hoax"

The third grep experiment documented here is "'Clear Eyd Fox Quickn Brown Hoax': A grep of the 1995 Poetics Logs". This text was generated in an effort to use the grep procedure to create a collaborative text. To this aim, what could be better, it was thought, than an entire year of discussion on the nation's premier electronic discussion list for innovative poetries. This grep uses the 1995 archive of the discussion list as the source text, a considerably large source text from

which to draw. The procedure for this work is somewhat more complicated than the grep works mentioned above, since the volume of grep results required editorial decisions to be made. As with all procedural works, it is crucial that editorial decisions are documented. They are presented here as details of this procedure.

"Clear Eyd Fox Quickn Brown Hoax" was generated by taking the phrase "Clear Eyd Fox Quickn Brown Hoax", my own variation of the old typewriter repairman's infamous "a quick brown fox jumped over the lazy moon", and using these words as pattern words. The work was generated and built within the UNIX C shell. For the first web version of this work, HTML was used to display the output files. For the print version, word wrap was determined by Netscape's default display parameters. (Internal truncation of title words was also guided by Netscape's pixel width on a standard monitor.)

Procedure: The UNIX grep command was used to parse the Poetics log files for 1995. Thus, every word in this text was written by persons posting to Poetics (that is, one or more of the approximately 200 subscribers to Poetics at that time). The input string was altered in some cases ("browning" for "brown," for example, as detailed below) to keep the output within the 3 to 100 line range which was deemed optimal for this composition.

Text Preparation & Images: Each section represents the unedited output for the grep returns on the indicated expression with two exceptions. (1) The "clear" section was truncated at 41 lines due to an excessive length. (2) Multiple repetitions of header lines were trimmed in the case of repeats due to hits based on threads. Images were a crucial component to the Web version of this work. For images, Alta Vista was used to search the Web based on search phrases identical or similar to the grep pattern word. The first usable (i.e., non-proprietary) image encountered was downloaded and incorporated into the installation.

The Input Phrases: Details about the input phrases are as follows:

Title Word	**Grep Pattern Word**	**Alta Vista Search Phrase (for Images)**
clear	clear	clear images
eyd	eyed	eye image
fox	fox	fox
quickn	quicken	quick images
brown	browning	brown
hoax	hoax	hoax images

Note: for images, "search word" + "images" was first tried (to force retrieval to contain image files). If this did not work, "search word" + "image" was tried. As a last resort, if nothing could otherwise be found, the search word on its own was tried, even though it meant looking through hundreds of hits. Interestingly, "fox" was the most difficult word to search because of numerous movie and sports hits containing proprietary material.

Date of Generation: "Clear Eyd Fox Quickn Brown Hoax" was generated on 12 November 1996.

Works Cited

Bernstein, Charles. "Making Words Visible." *The L=A=N=G=U=A=G=E Book.* Ed. Bruce Andrews and Charles Bernstein. Carbondale, IL: Southern Illinois UP, 1984. 284-286.

Drucker, Johanna. "Process Note: Marshall Reese, *Writing*." *The L=A=N=G=U=A=G=E Book*. Ed. Bruce Andrews and Charles Bernstein. Carbondale, IL: Southern Illinois UP, 1984. 263-266.

Heath, Steve. *Newnes UNIX Pocket Book*. Second ed. Oxford: Newnes, 1994.

Pastior, Oskar. "Poempoems." *The Printed Head* 1.5 (1990): 1-37.

Silliman, Ron. *The New Sentence*. New York: Roof Books, 1989.

Topham, Doublas W. *Portable Unix*. New York: John Wiley & Sons, 1992.

Notes

1 Grep is an instrument that enters the file to create a separate entity, delivering it to standard output. Grep gives such an emphasis to process that its default destination is not even a file, only an impermanent display on the screen.

2 Commands such as grep present interesting alternatives to hyper-dominant paradigms for electronic writing, such as those marketed by Microsoft. Curiously, the graphical environment that typifies most word processing programs is highly imprecise. User vulnerability to predictable aspects increases wherever compiled/proprietary code restricts the user from access to the source code of the word processing software. In UNIX systems, by contrast, access to source code is common.

3 Again, emphasis is on the unit of the line.

4 Though short-lived (1850-52), the friendship between Nathaniel Hawthorne and Herman Melville was an intense one. It began with each writer writing about the other's books and ends most markedly with Hawthorne in Melville's books as Vine in *Clarel* and as the subject of "Monody". It was Hawthorne's influence that may have turned *Moby Dick* from a sea adventure into a darker investigation. It was the idea that interactions between texts could so drastically alter their outcome that made these interesting as source documents.

5 None of Hawthorne's letters to Melville seem to have survived.