SLAC-PUB-3422
August 1984
(M)

AID - Access to Informal Documentation*

Roger B. Chaffee
Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

Author's current address:
Metaphor Computer Systems
2500 Garcia Ave.
Mountain View, California 94043

ABSTRACT

In many cases, access to stored information requires prior knowledge of the subject. It is a common joke among schoolchildren that to find the spelling of a word in a dictionary, you have to know how it's spelled.

A similar problem affects the documentation for computer programs and operating systems: before you can use even the best documentation for the "REPHALGAMIZE" command, you have to know that it exists and applies to your situation. If "rephalgamize" is not a word that springs to mind, you may never discover the existence of that command.

An experimental program which addresses this problem has been in use for the last four years on the large IBM timesharing system at the Stanford Linear Accelerator Center ("SLAC"). The program, called "AID", has been enthusiastically received, and has generated some interesting questions as well as providing many suggestions and answers.

(To be presented at ACM Special Interest Group on University and College Computing Services User Conference XII, Reno, Nevada, November 11-14, 1984)

* Work partially supported by the Department of Energy under contract DE-AC03-76SF00515 and, in part, by Metaphor Computer Systems.

1. The User's View

AID is invoked by typing 'AID *keyword*', where *keyword* is a word associated with some task or problem. For instance, the on-line documentation for AID gives the example 'AID DISASTER'. Typing 'AID DISASTER' produces

```
Try...
        HELP RECOVER
        SPACE ? RETURN
        SPACE TEMP Q RETURN
        HELP ARCHIVE
```

This result shows the main purpose of AID, which is not to provide information about commands so much as to provide information about information. The commands suggested in this example accomplish nothing by themselves, but they provide information about other commands. The command 'HELP RECOVER' gives information about using SLAC's temporary backup system, which might contain a recent version of an accidently destroyed file. The 'ARCHIVE' command, on the other hand, deals with permanently archived files. Finally, the 'SPACE' commands deal with entire 'minidisks' of temporary files, which are all too easily released to the system. Given this list, someone who has just destroyed a file or released a temporary minidisk may be able to find out what to do to get it back.

As in this example, it usually happens that most of the responses from AID are unrelated to the problem at hand. The database is built to give any information that has any chance of being appropriate, because of the chance that a wild guess will provide exactly the right suggestion.

2. The Program

AID has three major components: a database, a program to scan it, and a log to record usage. The program is not complex, and its exact form is not important to this discussion.

The entries in the database are lines of text, which can be edited by any Text Editor. They have the form

Message to be typed / KEYWORDS

For instance, one line in the AID database is

HELP RECOVER / RECOVERY DISASTER RESTORE GETBACK BACKUP
ARCHIVE LOST UNERASE UNDELETE RETRIEVE RETREIVE

The program scans every line in the database, looking beyond the ' / ' for words which begin with the word or characters typed by the user. If a match is found, the line is listed, up to the ' / '.

The third part, which is not seen by the user, is that every AID command is logged, that is, written to a file where it can be reviewed. Logging was put in as an afterthought, but has emerged as a critical part of the system. The logfile may be viewed only from the account which maintains the database, which provides a moderate level of protection and privacy.

3. Database Generation

The original database for AID was generated by "brute force". A list of all the possible "HELP" commands on the SLAC system was generated by dumping lists of the names of files containing "HELP" information, and augmented by listing the names of EXECS which were available to the general user, but might not be documented through the HELP system. No subroutine or procedure calls were included originally, although some have been added in the maintenance process. The original list contained about 400 commands.

The second part of generating the database was to go through the list of "HELP" commands and other responses that AID could produce, and to append the keywords. This was done mainly by a process of "free association", in which the author tried to think of any words that a user might use in a situation which could involve the given command. The original pass for this process took a couple of days, and was a lot of fun to do.

Once the initial database existed, so that AID could produce useful results in many cases, the feedback from the maintenance process was (and still is) important for adding and tuning.

4. Keyword Matching Algorithm

In order to reduce the number of keywords in the database, the user's word is processed before matching. All characters are converted to upper case, and trailing 'ING', 'S', and 'ES' are removed, unless the result would be shorter than three characters.

The resulting word matches a keyword in the database when the keyword in the database starts with that word. Thus, 'AID DIS' will list results for AID DISASTER, AID DISCOVER, AID DISABLE, AID DISPATCH, AID DISASSEMBLE, AID DISK, and others.

Keywords shorter than three characters must match exactly. For instance, 'L' and 'LI' are valid abbreviations of the 'LISTFILE' command. The database entry for 'HELP LISTFILE' contains 'L', 'LI', and 'LISTFILE', so 'AID L' , 'AID LI', and 'AID LIS' all produce the right answer. A lot of the database is put together with "brute force" techniques like this. Elegance is desirable, but expedience rules.

Some keywords, such as 'PRINT', produce too much output. AID accepts multiple argument words, and will list an entry only if all the argument words match keywords in that entry. This allows reasonable treatment of commands like 'AID PRINTING SCHEDULE'.

5. Database Maintenance

A few results have surfaced that were not foreseen when the AID project was started. Some of them are probably general to any installation, and some are specific to the SLAC environment.

- *The database should be updated frequently.*

  The computing environment at SLAC changes continually. An important part of the user's toolbox is programs which are developed by other users, and made available for public use. Announcement, documentation, and maintenance of these programs is informal, and AID has become an important branch in the local grapevine of information.

  One of the features that provides very good public relations, for AID and for the people who maintain it, is that it can provide recent information . This includes pointers to recent announcements and added commands, and also quick updates to handle new keywords. Various schemes were considered in the beginning of the project to improve performance and efficiency by compiling the database 'source' into hashed keywords, indices into a random-access message file, and the like. The author is now convinced that any added complexity would result in less frequent updates, and would reduce AID's usefulness.

- *The commands should be tested.*

  One of the more annoying moments in using a system for on-line documentation is trying to find the right syntax for a "Help" command so you can ask about the syntax for the command you're interested in. Typing a command suggested by AID should never result in 'SYNTAX ERROR' or 'UNRECOGNIZED COMMAND'.

- *The customer is always right.*

  Every request is a valid one, and is someone's best attempt at getting some information, even if the real question being asked is "Does AID work?" Common misspellings are handled, usually by including the misspelling in the database as well as the correct one. For instance, 'AID RETREIVE' produces the same responses as 'AID RETRIEVE', as you can guess from the listing above of the entry for 'HELP RECOVER'.

  This is especially important when someone tries AID for the first time. Both naive and experienced users, although for different reasons, want to believe that computers are difficult to use. When AID is first installed on a system, people will commonly try one frivolous command as a test, discover that it doesn't work, and never use AID again. In order to accommodate the new user, responses have been put into the database for commands such as "AID AID", "AID HELP", "AID ME", "AID GARBAGE", and the like. These don't provide useful information, but they do provide reassurance, and may induce the user enough to try AID with a real request.

- *A sense of humor helps.*

  A successful response for an obscure keyword is satisfying for everyone concerned. Even the wrong response may provide some useful information or at least a welcome change from the usual blank screen. "AID TIME", for instance, gives the system date and time commands, commands for evaluating program execution, and "POPCORN", which is the telephone number (767-2676) of the local telephone company's time-of-day message.

At one time a list of obscenities was included in the program (in encoded form, of course), and instead of replying 'No AID found for BLASPHEMY' or the like, AID would simply make no response at all. This "cute" feature disappeared during some change in the keyword matching scheme, and has never been put back.

- *The database requires feedback from the logfile.*

As mentioned above, all requests to AID are recorded. Part of the maintenance process is to review the recent requests, especially any which didn't match any keyword in the database. People are surprised and impressed when they try a failed request again a day later, and get a useful response.

The author did the maintenance of the database, usually as a 10- to 30-minute editing session, first thing in the morning, every morning. It was rarely a routine job. News items, public announcements and rumors were examined and included. The logfile was examined, and requests for which AID had not provided a response were scrutinized. Some were obvious misspellings or mistypings, and were ignored unless they seemed to be a recurring case. Others, especially in the beginning of the project, were simply new words that hadn't been used in the database. These became the subject of an informal review (something like "Hmmm, I wonder what was meant by that"), and the keyword was added to the appropriate line or lines in the database. One source of information in this case was the entire sequence of requests which a person used in a situation, rather than the isolated request for which AID provided no information.

Occasionally, a new keyword would trigger a search for more information, in the system documentation, from the system programmers, or from users working in the area suggested by the keyword. In response, a new line of information might be added, with the keyword that triggered the search and any other keywords that seemed appropriate.

6. AID at other installations

The current size of the AID database at SLAC is about 1350 lines. AID is invoked from 100 to 200 times a day, on the SLAC system which typically supports 150 to 250 interactive users during prime time.

The IBM version of AID has been copied to a few other VM/CMS systems. On the only system of which the author has recent knowledge, usage has been light, although a few people use it frequently.

A new version is running on a VAX/UNIX system, where the author uses it as an *aide-memoire* instead of wading through the UNIX documentation. It duplicates some of the function of the "apropos" command, but is more flexible and can respond to more keywords. Although it is publicly available, it has not been heavily publicized, and is never used except by the author.

The difference between the heavy use of AID at SLAC and the infrequent use elsewhere has not been adequately explained. Perhaps unsophisticated users are afraid to experiment, and expert users already have the knowledge they need to get their jobs done. The SLAC users are physicists who are accustomed to working with experimental equipment and imprecise information, and are unusually willing to accept the same limitations in their computer tools.

Also, there is a critical level of responsiveness, which must be reached before AID becomes a tool rather than an experiment. If AID fails too often to provide the right information, then people see it as useless, and they don't use it. In this case, there is no input to the database through the logfile, the only additions come from the imagination of the implementer, and AID will continue to fail. On the other hand, if AID succeeds enough for people to see it as useful, then they will use it. The way that they use it provides input for the database, and usage and responsiveness will grow together. The trick is to reach the level of positive feedback before the users decide that AID is useless, and to stay there by following the expanding services of the system and the changing needs of the users.

7. Extensions

The author is now working in a different company with different objectives, and plans no more work with AID.

Paul Kunz of SLAC has implemented a version of AID which uses a pointing device, in this case the screen cursor, to utilize the list of suggestions as a menu. A suggested command can be issued to the system by pointing to it, without re-typing. The author has used this version, and found it to be very "friendly", and definitely an improvement over the original.

AID could recognize the current computing environment, and use it as some sort of added keyword. This would have the very desirable effect of reducing the output from AID, which is sometimes more than one screen's worth.

On the IBM system, the user's environment is defined by the "minidisks" to which he or she has access at the moment. The AID database of the moment is distributed over all files named "AID KEYS", on any accessed minidisk. The system minidisk contains the AID program and the general database, but special-purpose minidisks, containing programs used by only a small group of users, can contain files with information which is used by AID only when the particular minidisk is accessed. An AID KEYS file on a private minidisk could contain personal information such as telephone numbers.

An analogous function on a UNIX system searches the user's home directory and any directories specified in the "PATH" environment variable.

Another possible extension is *required* keywords. That is, a word in a database line could be marked so that the line is listed only if that particular word is one of the arguments in the AID command. Other Boolean combinations could be defined as well. Again, the object would be to restrict the output and reduce the number of lines sent to the user.

A form of required keywords has been implemented in the UNIX version, which allows database lines of the form

```
Message / KEYWORDS / MORE KEYWORDS
```

The text is listed only if the arguments of the user's request match at least one keyword in each keyword group. This seems to be a flexible and general approach and is a superset of the current scheme so that the current database will continue to work in the same way. It was added recently, and has not been evaluated in practice.

8. Summary

AID is a database of text and keywords, a program to search it, and a logging procedure. It prints the appropriate text when one or some of the corresponding keywords are specified by the user. Maintenance of the database is simple and convenient, and the logging enables a short feedback path between the users and the maintainer. Because of the technical and logical simplicity of the system, it can be maintained easily and adapted to the needs of the local user community.

Acknowledgements

I am indebted to Linnea Dunn, who made valuable suggestions. I have also benefitted from Terry Schalk's unflagging enthusiasm as well as his good sense, and from the support and suggestions of the entire SLAC community.