Name That Char

David M. Tilbrook

No fixed address

ABSTRACT

The last year witnessed two strange but equivalent experiments in bars, one in Italy in April and the other in Atlanta in June. Rather than the normal concentration on the task of emptying glasses, the two to three dozen participants in each location were fervently scratching their heads and frequently exclaiming “This is harder than I thought!”, “Are you sure there are 33?”, or “Oh mi gawd ... how could I forget that one?” This paper poses the question that has stumped nearly a hundred UNIX® conference attendees and makes some (less than profound) observations on the problem and why it poses such a challenge.

The ASCII Character Challenge

The characters in the ASCII set should be as familiar to UNIX users as are the characters in “Thomas the Tank Engine” to the parents of a three or four year old. They are used each and every day, sometimes multiple times within a single line. Each has a number of unique characteristics [sic] and associations for each user, although the associations might be based on the shell or editor commonly used by the user. Yet when challenged to name all the members of the ASCII printable character set I have only heard of or observed one person who did it without pause and only one other who completed the test within five minutes. When the question is initially posed, nearly every participant responds with:

> “No problem”

What they mean is they’ll have no problem getting about 85 percent of them, but the remaining characters will take much longer than initially expected. So, without further ado, I will pose the question and give you the opportunity to conduct the experiment yourself.

The ASCII character set consists of 128 characters (i.e., octal 0 through to and including 0177). The thirty two lowest characters (octal 0 through 037) including the tab, newline, and carriage return, are commonly referred to as the control characters. The last character (0177) is the delete character. This leaves ninety-five other characters. Fifty-two of these are the alphabetics and ten are digits. This leaves thirty-three other characters in the ASCII set. The challenge is to name them all. But first of all some suggestions and rules.

- Only tools allowed are a piece of paper and a pen. For example, using a keyboard is definitely cheating. Jaap is allowed to use a crayon.
- The test should be conducted individually, although group efforts do not tend to be much more successful.
- When writing down the characters, do it in such a way as to facilitate quick counting (rows of eight seems the most common and convenient choice) because you’ll be counting them repetitively as you won’t believe you are missing as many as you are.
- Do give yourself at least five minutes to get the last one or two characters, but it is allowed for observers to give obtuse clues such as “Ever use the shell?”
- When you get to thirty-three check that you do not have any repetitions, as this has been frequently observed.

- Do try the test again even if you were subjected to it in Florence or Atlanta as you will be surprised how difficult it still is (i.e., you might remember that last elusive character; then again you might not).

Anyway, here is the challenge, as phrased to the other victims, er, contestants:

> • the ascii character set consists of 128 characters:
> • the first thirty-two are control characters,
> • the last is DEL,
> • sixty-two are alphanumerics,
>
> this leaves thirty-three other characters.
>
> so in the style of so many self-teaching courses
>
> NAME THE 33 MISSING CHARACTERS
> TRY IT NOW ...

Not as easy as you thought, was it, or perhaps you were helped by the presentation of the test in a written form, an aid that the other victims did not have.

Now at this point I could break to a new page so as to not give any clues on the page containing the challenge, but rather than waste paper (any more than this paper is a waste) I'll discuss the history of the challenge and some observations.

The first time this test was attempted (to my knowledge) was on a British Rail train from Aldershot to Waterloo. There is a noticeable lack of computing facilities on British Rail trains and having no book or paper to read I decided to think about an extension to *qed*(1) (oh yes Virginia there still are line editor users around). *qed* uses most of the ASCII characters in some form or another and what I required was a character that was in some way a mnemonic and currently unused. The required operation was to specify a shell command through which addressed lines were to be piped and the output was to be appended to another buffer at a designated address. *

After a few minutes of thought I decided to write down all the punctuation characters so as to be able to eliminate the ones already used as commands or addresses. This exercise proved to be more difficult than I had expected and I recalculated repeatedly the number of characters to ensure that my initial evaluation of 33 was correct. The first 25 characters had come quickly, the next five or six more slowly and the penultimate one after a long pause. The last character came just as the train arrived and so surprised me when I got to the office, I gave the test to my first victim (Pat Place). He, to my amazement, had difficulty with the same characters. We then proceeded to give the test to every unsuspecting IST employee who wandered within our grasp. It was during this phase that we came across the only person to have completed the test (to our knowledge) without a pause, that being Jeremy Huxtable, a person who spent a lot of time dealing with 7000 Chinese characters, which were represented by combinations of two ASCII characters. Despite his discouraging performance, we were heartened to recognise that having a great deal of difficulty with the challenge is the norm, and not just our own inadequacy. We also observed at IST and later at LUUGACAS*, EUUG and USENIX meetings the same character was being left to the very end in a majority of the tests.

Oh My Giddy Aunt ... How could I forget that one?

The character that is usually the last to be found, is the humble space (octal 040). At this point in this paper, that character constitutes over 18% of the text entered, so its use should not be considered uncommon. Initially I thought the frequency of this oversight was due to the posing of the question, but the question is now stated to clearly include the space in the required set of characters. Furthermore, there is another indication that people tend to forget the space. Frequently the clue I give to help people when the

---

* *qed* has had the facility to pipe text through ('|'), to ('>') and from ('<') pipes since 1977 (before *vi*), however, the '|' removed the original input lines, which was some times not desirable. In fact, now I rarely use the '|' facility and almost always pipe selected lines through a command into an empty buffer.

* The London Unix User Group and Curry Appreciation Society which convenes the last Thursday of every month except December.

only character they are missing is the space is that they should think about the biggest key on the keyboard, to which everybody invariably responds “RETURN?” (didn’t we, Kirk?). I have yet to see, nor would I use, a keyboard on which the size of the return key approaches that of the space key (space KEY? ... what’s this space KEY? ... it’s space BAR!).

The second most difficult character according to our somewhat unscientific observations is the equals sign (‘=’). Many people go through the exercise of thinking of all the C operators (‘+’, ‘-’, ‘/’, ‘*’, etc.) but skip over the ‘=’ for some reason. I have no theories, but it is rather amazing how often this oversight occurs.

Other characters that people tend to frequently leave to the end are ‘~’ (even the csh users), ‘?’ (used in ed, sh, c, and English) and the ‘^’ (used to be equivalent to ‘|’ in *sh*(1)).

Another interesting observation is that long time UNIX users had no trouble with ‘@’ and ‘#’ (ah the good old days) whereas more recent users don’t appreciate those characters’ original importance.

Techniques Used

Many testees thought that the initial way to solve the challenge was to think about the keyboard, but that was usually abandoned relatively quickly. Your fingers know where the keys are, but your brain does not. Furthermore, you have to think of the character before your fingers can find them (“just what is that character over the 6?”). Also what keyboard are you using and how often do you switch? Have you found two keyboard manufacturers who agree on the placement of the ‘\’ and ‘|’, which one is shifted, and their placement on the same key? Furthermore, if this approach was truly useful, why was it so many people missed the bottom row (the space-bar).

The other approach is to think of all the operators in C, and that approach taken to its extreme does yield 23 characters. If one considers spaces, quotes, brackets and the ‘#’ sign as C operators the only unused characters are “$@‘_”.

Actually the most common successful technique was the pairing or grouping of characters as in “()<>[]{}”, and “.:,;”, but this, whilst helping people get to the plateau that seems to exist at about 28 to 30, doesn’t help with ‘^’ or ‘_’.

Conclusions

You have a lot of nerve expecting a conclusion for a paper as gratuitous as this one. I leave it as an exercise for the reader to determine the reasons their characters were ordered as they were and why they had difficulty with the last few.

For those of you who are still missing characters here’s a lexically ordered hint:

    !"#$%&’()*+,-./:;<=>?@[\]^_‘{|}~

Now for the next challenge, name all the APL operators.