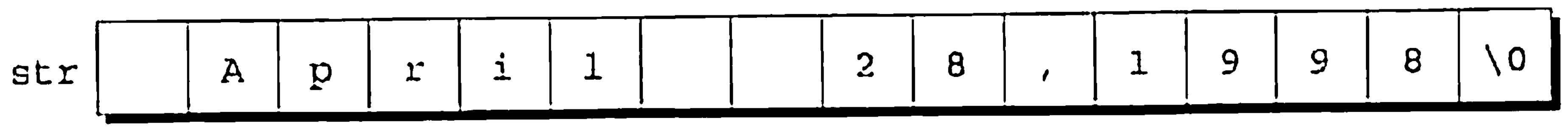
of the token by storing a null character in `s1` just after the last character in the token; it then returns a pointer to the first character in the token.

What makes `strtok` especially useful is that later calls can find additional tokens in the same string. The call `strtok(NULL, s2)` continues the search begun by the previous `strtok` call. As before, `strtok` marks the end of the token with a null character, then returns a pointer to the beginning of the token. The process can be repeated until `strtok` returns a null pointer, indicating that no token was found.

To see how `strtok` works, we'll use it to extract a month, day, and year from a date written in the form

month day, *year*

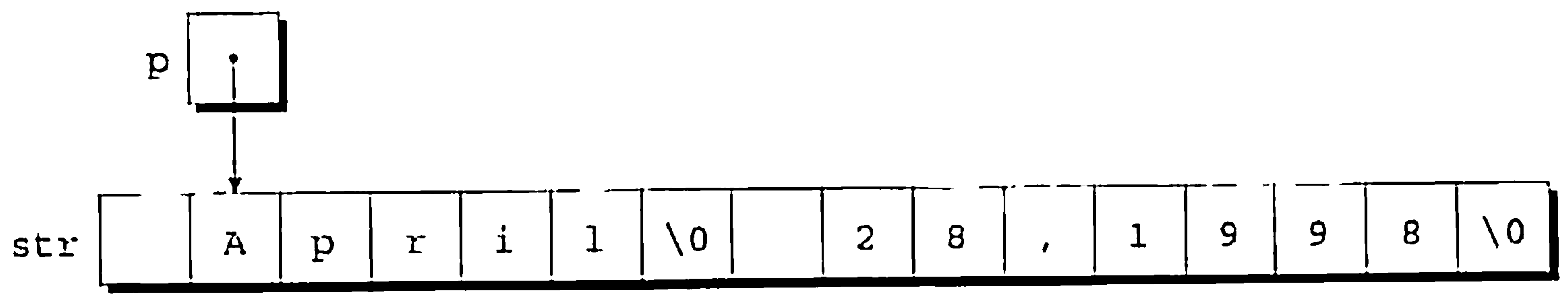
where spaces and/or tabs separate the month from the day and the day from the year. In addition, spaces and tabs may precede the comma. Let's say that the string `str` has the following appearance to start with:



After the call

```
p = strtok(str, " \t");
```

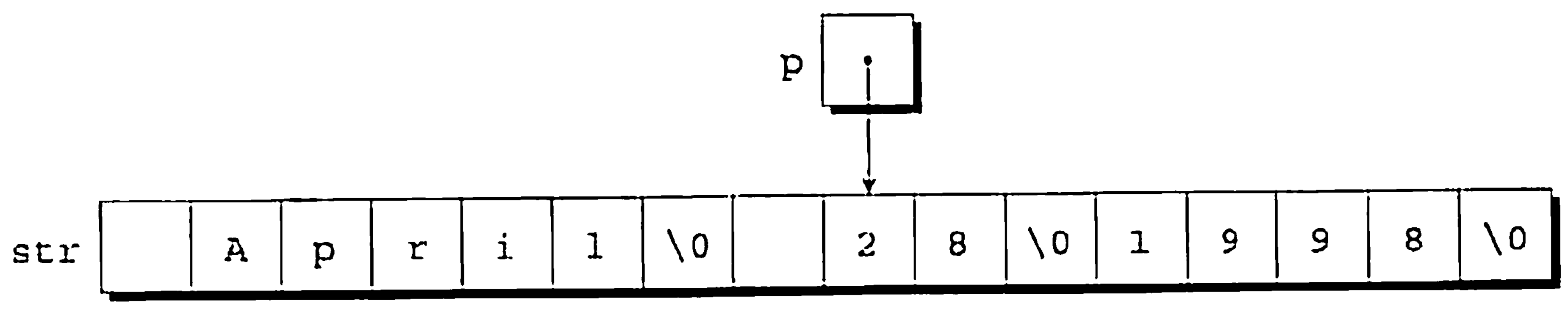
`str` will have the following appearance:



`p` points to the first character in the month string, which is now terminated by a null character. Calling `strtok` with a null pointer as its first argument causes it to resume the search from where it left off:

```
p = strtok(NULL, " \t,");
```

After this call, `p` points to the first character in the day:



A final call of `strtok` locates the year:

```
p = strtok(NULL, " \t");
```