Lab **xargv** — Extended Argument Vector

CS 2370

Background

You have learned how to receive *command-line arguments* by means of the arguments to **main** conventionally named **argc** and **argv**. In this lab you will *extend* **argv** by allowing special arguments that refer to files containing further arguments. Such a filename begins with the **@** character. You will collect all arguments in such files in the order you encounter them, and then return to where you left off. Such @-files can reference other @-files.

Requirements

To illustrate what needs to be done, suppose the following files have the contents indicated:

File *argfile.dat*:

```
we will sell
no soda @argfile2.dat before
its
time
```

File *argfile2.dat*:

```
now is the time
for all carbon-based
units @argfile3.dat to come to the aid of
their sector
```

File *argfile3.dat*:

```
where no one has gone
before
```

Your task is to collect all arguments in the order they are referenced by opening the @-files as you encounter them. For example, if your program is named **xargv**, the following command line

```
$ xargv one @argfile.dat two
```

would print the following results:

```
32 items:

one
we
will
sell
no
soda
now
is
the
```

```
time
for
all
carbon-based
units
where
no
one
has
gone
before
to
come
to
the
aid
of
their
sector
before
its
time
two
```

Whenever a regular string is encountered it is appended to the vector. Whenever an @-file is encountered, it is immediately opened and the process continues inside that file. This *nesting* process can go *arbitrarily deep*. When an @-file completes, the process picks up where it left off before opening that file. Note that lines in the text file are not important here; you just read space-delimited strings wherever they are in the text files.

Write a program that works as described above, printing the argument strings collected to standard output.

Implementation Notes

To have a procedure that repeats in different contexts, you need some sort of repetition that can go on without a pre-arranged limit. So, a **for**-loop alone won't quite do the job. *Hint*: you might consider recursion somewhere in your thinking.