File and Glob iterator proposal

August 4, 2014

The goal of this proposal is to show the interface that we are thinking of implementing (we being Brad and Tim). The intention of this iterator is to yield the directories and or files reachable from a given directory and is meant to model in some way, the semantics of `glob`, `wordexp`, and `listdir` (and functions along these lines).

1 User facing interface

The interface to the user would be the following:

```
iter glob(pattern   : string = "",
          startdir  : string = "",
          recursive : int    = 0,
          files     : bool   = true,
          sorted    : bool   = false,
          dirs      : bool   = false,
          expand    : bool   = false,
          dotfiles  : bool   = false) : string
```

where

- `pattern` is a valid `glob` or `wordexp` glob pattern. More information on what exactly a valid glob pattern (or wordexp pattern) is, can be found on the `man(3)` page for `glob` and `wordexp`.
- `sorted` tells us whether we should return a sorted output. These would be sorted at the directory level, and using the *full path* (starting from `startdir`) of the string yielded from te iterator.
- `dirs` Says whether we should yield directories or not.
- `files` Says whether we should yield files or not.
- `expand` Says whether we should expand environment variables, or run shell commands in `pattern` (a la `wordexp`). For more information on this, see the `man(3)` page for `wordexp`.
- `startdir` directory to start searching from.

- `recursive` tells us whether we should be recursive or not. If `recursive` is set to 0 we do not recurse into subdirectories, if `recursive` is set to -1 we recurse into all subdirectories. Otherwise, we recurse to a depth of `recursive` num. Whether or not we should do this or use a `bool` instead is an open issue. (see **??**)

- `dotfiles` Should we yield dotfiles or not? By default, we do not yield dotfiles (or recurse into dot directories).

We would (of course) also expose a parallel version of this iterator.

Other, additional flags have been discussed for the possible future. For a discussion on what these are see **??**.

The general thinking that we had when designing this interface was the following: *I want to...*

- "iterate over dirs" → `glob(dirs=true, rec=<num>)`
- "give me everything" → `glob(rec=<num>)`
- "give me wordexp" → `glob(pattern=..., expand=true)`
- "give me C glob" → `glob(pattern=..., sorted=true)`
- "give me Python glob" → `glob(pattern=...)`
- "give me command-line glob" → `glob(pattern=..., sorted=false)`

2 Open Issues

The following are flags that we have discussed possibly putting in, however we are seeking opinions as to whether or not we should add these in.

- Is there a better name for this iterator other than `glob`. (We're open to suggestions).

- Support sorting for parallel invocations of the iterator. Since, in this case we would either have to synchronize at the directory level, or sort everything all at once.

- A flag to drop the final slash when yielding directories.

- A warning or verbose flag. The idea behind this flag would be to warn you about certain underlying problems. For instance, if we were using `glob` with `expand = true` and `pattern = "$CHPL_HOME/*"` and we had not set `$CHPL_HOME`, we would generate a warning that we were using an undefined environment variable in expansion.

- Add a `skipdirs` flag. The idea behind this being that if we wanted to avoid certain directory or file names (*e.g.* `.svn` or `.git`), we would set this flag to the string we wanted to skip.

- Add a `symlinks` flag to say whether or not we should follow symbolic links. The main problems that we see with are (1) it add another flag, and (2) we could get ourselves into a possible infinite loop if a symlink points to a directory that contains the directory we are currently in.
- Should, `recursive` be a boolean. In this case we have two possible proposals:
  - We simply support recursive or non-recursive searching. The depth to which we should recur is not specifiable.
  - We add a `depth` flag that specifies the depth that we should recur to. (defaults to 0)

3 Examples

3.1 Example 1

```
// Yield all subdirectories of the current directory
// Won't recurse (so we only get the children of of the current dir)
// think: ls -l | egrep '^d'
for dir in glob(dirs=true, files=false) {
  writeln(dir);
}
```

3.2 Example 2

```
// Yield all subdirectories of the current directory
// In this case we get all subdirectories
// think: ls -l -R | egrep '^d'
for dir in glob(dirs=true, files=false, recursive=-1) {
  writeln(dir);
}
```

3.3 Example 3

```
// Yield all subdirectories of the current directory
// In this case we get all subdirectories up to depth 10
for dir in glob(dirs=true, files=false, recursive=10) {
  writeln(dir);
}
```

3.4 Example 4

```
// Yield all files in the current directory
// think: ls -l | egrep -v '^d'
for fl in glob() {
  writeln(fl);
}
```

3.5 Example 5

```
// Yield all files in and children of the current directory
// think: ls
for fl in glob(dirs=true) {
  writeln(fl);
}
```

3.6 Example 6

```
// Yeild all .c files in the current directory
// think: ls *.c
for fl in glob(pattern="*.c") {
  writeln(fl);
}
```

3.7 Example 7

```
// Yield all files in the current directory in sorted order.
for fl in glob(sorted=true) {
  writeln(fl);
}
```

3.8 Example 8

```
// Yield all .c files from this directory down
// think: ls -R *.c
for fl in glob(pattern="*.c", recursive=-1) {
  writeln(fl);
}
```

3.9 Example 9

```
// Yield all [a-p] directories from this directory down
// think: ls -R [a-p]
for fl in glob(pattern="[a-p]", dirs=true, recursive=-1) {
  writeln(fl);
}
```

3.10 Example 10

```
// Yield all [a-p] directories from this directory down. Yield them in sorted order.
for fl in glob(pattern="[a-p]", dirs=true, recursive=-1, sorted=true) {
  writeln(fl);
}
```

3.11 Example 11

```
// Yield all [a-p] directories from the $CHPL_HOME directory down. Yield them in sorted order.
for fl in glob(pattern="$CHPL_HOME/[a-p]", dirs=true,
               recursive=-1, sorted=true, expand=true) {
  writeln(fl);
}
```

3.12 Example 12

```
// Yield all [a-p] directories from the $CHPL_HOME directory down. Ignore all
// .git directories and files. Yield them in sorted order.
for fl in glob(pattern="$CHPL_HOME/[a-p]", dirs=true,
               recursive=-1, sorted=true, expand=true, skipdirs=".git") {
  writeln(fl);
}
```