```
'''
This is a skeletal working web spider, with virtually no error-checking.
You need to pass in an URL that points to a directory (e.g.
http://www.foo.com/bar/).

This version adds brute-force threads, spawning and killing one thread per
retrieve operation (instead of reusing threads).  That means this script
can run afoul of the 1.5.2 multi-CPU thread bug.  In addition, the main loop
uses polling on thread completion, which is inefficient.
'''

import sys
import string
import urllib
import urlparse
import htmllib
import formatter
from cStringIO import StringIO
import threading
import time

MAX_THREADS = 3

class Retriever(threading.Thread):
    def __init__(self, URL):
        threading.Thread.__init__(self)
        self.done = 0
        self.URL = URL

    def run(self):
        print "Retrieving:", self.URL
        self.page = urllib.urlopen(self.URL)
        self.body = self.page.read()
        self.page.close()
        self.parse()
        self.done = 1

    def getLinks(self):
        return self.parser.anchorlist

    def parse(self):
        # We're using the parser just to get the HREFs
        # We should also use it to e.g. respect <META NOFOLLOW>
        w = formatter.DumbWriter(StringIO())
        f = formatter.AbstractFormatter(w)
        self.parser = htmllib.HTMLParser(f)
        self.parser.feed(self.body)
        self.parser.close()


class Spider:
    def __init__(self, startURL, maxThreads):
        self.URLs = []
        self.queue = [startURL]
        self.URLdict = {startURL: 1}
        self.include = startURL
        self.maxThreads = maxThreads
        self.numThreads = 0
        self.threadList = []

    def checkInclude(self, URL):
        return string.find(URL, self.include) == 0

    def run(self):
        while self.queue or self.threadList:
            while self.queue and (self.numThreads < self.maxThreads):
                URL = self.queue.pop()
                self.getPage(URL)
            self.checkThreads()
        self.URLs = self.URLdict.keys()
        self.URLs.sort()

    def checkThreads(self):
        tmpNumThreads = self.numThreads
        for ret in self.threadList[:] :
            if ret.done:
                self.processPage(ret)
                self.threadList.remove(ret)
                self.numThreads = self.numThreads - 1
        if tmpNumThreads == self.numThreads:
            time.sleep(1)

    def getPage(self, URL):
        ret = Retriever(URL)
        ret.start()
        self.threadList.append(ret)
        self.numThreads = self.numThreads + 1

    def processPage(self, page):
        for link in page.getLinks():
            # Handle relative links
            link = urlparse.urljoin(page.URL, link)
            print "Checking:", link
            # Make sure this is a new URL and is within the current site
            if ( not self.URLdict.has_key(link) ) and self.checkInclude(link):
                self.URLdict[link] = 1
                self.queue.append(link)


if __name__ == '__main__':
    startURL = sys.argv[1]
    spider = Spider(startURL, MAX_THREADS)
    spider.run()
    print
    for URL in spider.URLs:
        print URL
```