How to do it...

Let's write a Bash script with the help of the `curl` command to find out the broken links on a web page:

```
#!/bin/bash
#Filename: find_broken.sh
#Desc: Find broken links in a website

if [ $# -ne 1 ];
then
  echo -e "$Usage: $0 URL\n"
  exit 1;
fi


echo Broken links:

mkdir /tmp/$$.lynx
cd /tmp/$$.lynx

lynx -traversal $1 > /dev/null
count=0;

sort -u reject.dat > links.txt

while read link;
do
  output=`curl -I $link -s | grep "HTTP/.*OK"`;
  if [[ -z $output ]];
  then
    echo $link;
    let count++
  fi
done < links.txt

[ $count -eq 0 ] && echo No broken links found.
```

How it works...

`lynx -traversal URL` will produce a number of files in the working directory. It includes a file `reject.dat`, which will contain all the links in the website. `sort -u` is used to build a list by avoiding duplicates. Then, we iterate through each link and check the header response by using `curl -I`. If the header contains the first line to have `HTTP/1.0 200 OK` as the response, it means that the target is not broken. All other responses corresponding to the broken links are printed on the screen.

> From its name, it might seem like `reject.dat` should contain a list of URLs, which were broken or unreachable. However, this is not the case, and lynx just adds all the URLs there.
>
> Also note that `lynx` generates a file called `traverse.errors`, which contains all the URLs that had problems in browsing. However, `lynx` will only add URLs which return `HTTP 404 (not found)`, and so we will lose other errors (for instance, `HTTP 403 Forbidden`). This is why we manually check for statuses.

See also

- The *Downloading web page as formatted plain text* recipe in this chapter explains the `lynx` command
- The *A primer on cURL* recipe in this chapter explains the `curl` command

Tracking changes to a website

Tracking changes to a website is helpful to web developers and users. Checking a website manually in intervals is really hard and impractical. Hence, we can write a change tracker running at repeated intervals. When a change occurs, it can play an audio or send some other notification. Let us see how to write a basic tracker for the website changes.

Getting ready

Tracking changes in terms of Bash scripting means fetching websites at different times and taking the difference by using the `diff` command. We can use `curl` and `diff` to do this.