badnodes

COLLABORATORS			
	TITLE : badnodes		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 6, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

badnodes

1.1 BadLinks v1.15 ©1993 by Roger Nedel

BadLinks:
========

1.2 BadLinks v1.15 ©1993 by Roger Nedel

Written By

Roger E. Nedel
3635 East Avondale Dr
Salt Lake City, Utah 84121-5504
U.S.A.

(801) 944-4352

1.3 BadLinks v1.15 ©1993 by Roger Nedel

Concept:
=======

I just finished authoring a HUGE amigaguide file, and was faced with the unenviable task of testing each and every button in the guide.

I was concerned that I might have mistyped some of the links. If that had occurred, then some unsuspecting user would attempt to access a node, only to find that the button wasn't linked properly.

After about 15 minutes of button pressing, I decided to write a program to do the checking for me...after all, I'm pretty smart, aren't I???

1.4 BadLinks v1.15 ©1993 by Roger Nedel

Installation:
============

There's not much to it. Just place it in the desired drawer and you're ready to go.

1.5 BadLinks v1.15 ©1993 by Roger Nedel

Operation:
=========

BadLinks will run from the CLI only. I'll break program operation down into two sections.

Testing a file with NO external links

Testing a file with external links

```
                What is an external link?
========================
```

   In case you're scratching your head, wondering what the
   heck an external link is...

   Amigaguide files generally access nodes which lay solely
   inside their own files. If this is true of your amigaguide
   file, then you don't have any external links.

   An amigaguide file can access a node inside another
   amigaguide file. If this is the case, then your file
   contains external links.

1.6 BadLinks v1.15 ©1993 by Roger Nedel

```
Testing a file with NO external links:
====================================

If your file contains NO exernal links, then operation of
"badlinks" is very simple.  The syntax of it's CLI command
is as follows:

      badlinks <file>

where:   file = amigaguide pathname.

Badlinks will verify that every "link" attempt has a matching
"node".  You can expect badlinks to generate the following
files:

      Ram:NodeNames
      Ram:InvalidLinks


Ram:NodeNames:
=============
```

   This file is created only if your amigaguide file contains
   one or more valid nodes. If it doesn't contain any nodes,
   then "badlinks" will tell you so, and will refrain from
   creating this file.

   If this file is created, it will contain an alphabetical
   listing of each node name found in your amigaguide file.

```
Ram:InvalidLinks:
================
```

   This file is created only if your amigaguide file contains
   one or more invalid links. If "badlinks" can't find any

invalid links, it will inform you of that fact, and will
refrain from creating this file.

1.7 BadLinks v1.15 ©1993 by Roger Nedel

```
                Testing a file with external links:
==================================

If your file contains exernal links, then operation of "badlinks"
is a bit more difficult.  The syntax of it's CLI command is as
follows:


      badlinks <file1> [<file2> <prefix2>] [<file3> <prefix3>]...


where:   file   = amigaguide filename(s)

                prefix
                 = path to add to beginning of external links

         <>     = required argument
         []     = optional argument


Badlinks will verify that every "link" attempt has a matching
"node".  You can expect badlinks to generate the following
files:

      Ram:NodeNames
      Ram:InvalidLinks


Ram:NodeNames:
=============

   This file is created only if your amigaguide file contains
   one or more valid nodes.  If it doesn't contain any nodes,
   then "badlinks" will tell you so, and will refrain from
   creating this file.

   If this file is created, it will contain an alphabetical
   listing of each node name found in your amigaguide file.


Ram:InvalidLinks:
================

   This file is created only if your amigaguide file contains
   one or more invalid links.  If "badlinks" can't find any
   invalid links, it will inform you of that fact, and will
   refrain from creating this file.
```

1.8 BadLinks v1.15 ©1993 by Roger Nedel

```
Prefix:
======

   The most difficult thing to understand about "badlinks" is
   the "prefix" concept.  To be honest though, if you've created
   external links in an amigaguide, it won't be tough at all.

   If you have a node in file1 labelled <@node "internal node">,
   then the following link will tie into it:

         a{" Go internal " link "internal node"}



   Now taking it one step further, lets say you want to link into
   a node from ANOTHER FILE (file2).  Let say that the node in
   file2 is called <@node "external node">.  To link into this
   node, the following WILL NOT suffice:

         a{" Go external " link "external node"}



   Why won'that suffice.  Because amigaguide needs to have some
   sort of a path telling it which file the node belongs in.  For
   example, the following would work if both file1 and file2
   resided in the same drawer:

         a{" Go external " link "file2/external node"}



   This illustrates the prefix concept.

         a{" Go external " link "PREFIX/nodename"}



Examples:
========

Prim File:  work:help.guide

     Link:  a{" Mystuff " link "MY.GUIDE/main"}

 2nd File:  work:myguide

      CLI:  badlinks  work:help.guide  work:my.guide  MY.GUIDE

=====================================================================

Prim File:  work:help.guide

     Link:  a{" Mystuff " link "WORK:MY.GUIDE/main"}
```

```
 2nd File:  work:my.guide

      CLI:  badlinks  work:help.guide  work:my.guide  WORK:MY.GUIDE

=================================================================

Prim File:  ram:file1

    Links:  a{" Index " link "Index"}           /* internal link */
            a{" Go 2  " link "FILE2/main"}      /* external link */
            a{" Go 3  " link "WORK:FILE3/main"} /* external link */

 2nd File:  ram:file2
 3rd File:  work:file3

      CLI:  badlinks  ram:file1  ram:file2  FILE2  work:file3  WORK:FILE3
```