Dir

Dir with attitude!

'Dir' with that X factor!
Power user's directory listings in almost any format.
Astonishing set of features!
Full unix-like file-name matching.
Generate and execute batch scripts directly from the output.
Select files older than 'x' days or using almost any other criteria such as empty folders.
Search for duplicates using SHA1 or CRC32 digests.
List all files on your drive to track down those space-hungry folders.
Move/copy files to a new date-stamped archive folder.
Wildcard file renaming.
Execute the output as a batch script automagically. Even call XDir from within the script.
Sort output on any file attribute.
Too many features to list here.
I am still finding new uses for this tool myself!

History

Birth

XDir has been years in the making, as all good tools should. It started as a tool sent to all units in a very distributed network with various, often unreliable, methods of transfer of data files and code. Its simple aim was to generate a computer readable list of all data files and programs. The list was then pulled back for analysis to determine which units on the network had the correct versions of the components making up the system.
All it did, back then, was to list every file and directory, recording its name, creation date and time and size in bytes.

```
XDir Source\*.c
<Produces>
C:\Dev\XDir\source\Crc32.c,02/02/2006 00:32:22,4981
C:\Dev\XDir\source\Params.c,17/02/2006 23:29:34,27963
C:\Dev\XDir\source\Sha1.c,02/02/2006 00:32:60,10338
C:\Dev\XDir\source\Wild.c,17/02/2006 02:05:02,23855
C:\Dev\XDir\source\XDir.c,17/02/2006 23:45:12,32991
```

Toddler stages

As it developed, along with other tools, it became more stable and more flexible. More and more parameters were requested and provided. It was during this phase that the command-line parameter handling was generalised and re-used in several other packages.
Many features were added to the parameter handler. Among the most significant were the ability to read parameters from a text file, defining parameter values from the values of others and collecting simple data contained in data files.

```
XDir @Test.inp
<Test.inp>
: Root folder for search
/Root=\Dev\XDir\
: All Sources.
*.c
: All Headers.
*.h
: Walk subdirectories
/S
<Produces>
\Dev\XDir\Source\Crc32.c,02/02/2006 00:32:22,4981
\Dev\XDir\Source\Params.c,17/02/2006 23:29:34,27963
\Dev\XDir\Source\Sha1.c,02/02/2006 00:32:60,10338
\Dev\XDir\Source\Wild.c,17/02/2006 02:05:02,23855
\Dev\XDir\Source\XDir.c,17/02/2006 23:45:12,32991
\Dev\XDir\Include\bool.h,30/07/2004 11:39:48,859
\Dev\XDir\Include\CDefs.h,05/01/2006 16:31:46,8036
\Dev\XDir\Include\Crc32.h,16/09/2004 17:10:28,1109
\Dev\XDir\Include\Params.h,17/02/2006 11:26:52,4531
\Dev\XDir\Include\Sha1.h,16/09/2004 17:21:16,1122
\Dev\XDir\Include\Wild.h,09/02/2006 19:53:44,1507
```

Teenage years

XDir went through a normal angst ridden growth spurt, complete with spots and aggressive behaviour. Its parameter system allowed it to achieve more and more

complex functionality. Using it in a batch script to generate another batch file to call afterwards became a common and widespread advance. This provoked the creation of the /Form parameter and from here it grew into adulthood.

```
: Bkp.inp --- Backup all sources and headers to a dated backup folder.
: Call with:
: XDir @Bkp.inp>Bkp.BAT
: CALL Bkp.BAT
: DEL Bkp.BAT

: All Sources and headers
*.c
*.h
: Sub directories
/S

: Get the dir name.
: Today= format id dd/mm/ccyy so format selects ccyymmdd
/Bkp={/Today=[78904512]}Backup
: Make the directory if it doesn't exist.
/Form=if not exist {/Bkp=} md {/Bkp=}
: Copy the file. (` is converted into ")
/Form=copy `*D*P*F*E` {/Bkp=}\
```

The future

New feature planned include:
Wild character matching within the file path such as:

```
: All files whose path contains a folder called 'Temp'
*\Temp\*
```

Padding to left or right with any character. Use for padding numbers with zeros for sorting.

Parameters

How to specify them

Obviously, you can list parameters on the command line. You can list parameters in a file and include them with the usual @«File» mechanism. Parameters are differentiated from file masks in that they start with a '/' character.

INP Files

Once these parameters are processed, any file called "**XDir.INP**" in the local folder will be processed, then, if the location of **XDir.EXE** is different and contains an **XDir.INP** that will be read in too.
INP files have a simple structure. Blank lines are ignored. Trailing spaces are discarded, but not inside quotes. Lines starting with a ':' are also skipped so you can use them as comments.

#IF in INP files

The only unusual feature is the '**#IF**' system. If you start a line with '**#IF**', the next string is looked up in the parameters defined so far and only if that parameter is defined will lines following up to the end of the file or the next '**#IF**' be included. The opposite applies for '**#IFNOT**' lines, parameters following are only included if the specified parameter is not defined. You can use this feature to build libraries of useful parameter sets selectable by a single parameter at the start of the file or on the command line.

How they work

You can define parameters in any way you like. For flags I usually use names like /Long. These are either specified or not. For valued parameters I use a name, an '=' and a value, such as /Root=C:\Windows.

Groups

There are two groups of parameters. The global set and the variable set. The global set contains simple commonly used system environment values such as /Today= (today's date). The variable set is defined by you or the file currently being examined and can appear in any order.

Referring to each other

Parameters can refer to each other so long as those they refer to appear earlier. That's why the fixed set is created first, you can therefore always refer to them. This mechanism can be extended to use a parameter to select one of a number of other parameters. To access an existing parameter, enclose its name in {braces}.

```
: Define my month names
/Month01=January
/Month02=February
…
/Month12=December

: Define my day names
/Day01=First
/Day02=Second
/Day03=Third
…
/Day28=Twenty {/Day08=}
/Day29=Twenty {/Day09=}
```

```
/Day30=Thirtieth
/Day31=Thirty {/Day01=}

: Full date form
"/Form=*F*E,{/Day*d[12]=} of {/Month*d[45]=} *d[7890]"
>>>>>>>>>>
Crc32.c,Second of February 2006
Params.c,Seventeenth of February 2006
Sha1.c,Second of February 2006
Wild.c,Seventeenth of February 2006
XDir.c,Seventeenth of February 2006
```

Formatting

You can select parts of parameters with a format enclosed in '[]' brackets. The format is a sequence of digits and letters specifying which character to select. A '-' has the obvious effect. Use a '#' to suppress zeros. For example, since the /Today= parameter has the format dd/mm/ccyy (like all other dates), {/Today=[78904512]} will produce ccyymmdd in the output. Note that the characters are counted from 1 and 10 is indicated by 0. Further characters can be selected using 'A' for the eleventh character, 'B' as the twelfth etc.

Getting from file contents

You can set the value of a parameter from a file using the format `@«FileName»[place]`. The options available in the `[place]` part can take one of the following forms:

```
<Ln>     Include line 'n' from the file (maximum 80 characters). Lines are
         numbered from 1.
         E.g <L5> collects the 5th line from the file.
<n!f>    Offset and format.
         E.g. <4!%f> collects the float value starting at byte 5.
<n-m>    Collects characters starting at byte n and ending at byte m.
         E.g <5-15> collects the 10 characters beginning from byte 5.
<n:m>    Collects m characters beginning at byte n.
         E.G. <5:10> collects the 10 characters beginning from byte 5.
```

Remember that the file name can be the name of the file currently being printed. For example:

```
: Print the file name and the first line of the lile.
/Form=*f,@*f<L1>
```

What they are

«File mask»

The file mask is treated like the unix-like operating systems do. Searching for *abc* will find any file whose name contains the letters 'abc' in it. Remember that this means that "*.*" finds files with extensions and skips files with no extension.

Masks are case insensitive.

You can use as many file masks as you like. Each is treated separately as if `XDir` was invoked once for each one. When printed, folders finish with a '\' character, files don't.

@«File»

Includes parameters defined in the specified file. I usually use a '`.inp`' extension on my parameter files put that's only a personal preference.

Parameters in files appear one per line. Blank lines are ignored, as are lines that start with a ':' character.

Command line parameters

Command line parameters can appear in any order. They can refer to each other or global parameters in the usual way.

/S

Walk all subfolders.

/Dirs

List directories. Normally, XDir only lists files.

/NoFiles

Don't list files. Usually used along with /Dirs to list only directories.

/Long

List the long name of the file if possible. If the short form is not available, the long form is used by default so this parameter is rapidly becoming obsolete.

/Root=

Specifies a root path to base all searches from.
If you find yourself using something like:

```
XDir \Dev\XDir\Debug\*.obj \Dev\XDir\Release\*.obj
```

You can use instead:

```
XDir /Root=\Dev\XDir\ Debug\*.obj Release\*.obj
```

/INPPath=<Path>

Specifies an additional path to search for parameter files.

/Head=

Text to print at the start of the output. Repeat as required.

/Tail=

Text to print at the end of the output. Repeat as required.

/Form=

The form parameter defines the form of the output for each file found matching the mask. Forms often contain spaces, tabs, commas and pipe characters so surrounding the whole parameter in quotes is a good idea. See later for a detailed description of the structure of a /Form= parameter.

/Sort=

Sorts the output. Specify the sort order in the same style as the /Form= parameter. Multiple forms and multi-line output is handled correctly. If /Execute is specified, sort is performed before execution. /Head= and /Tail= output still appears at the start and end of the sorted output.
Use a '-' at the start of the spec to indicate reverse sorting.
Sorting is done using text comparison only so to sort by date for example, use {/ZDate=} as this value is zero padded. I will soon work on a format that will allow zero padding on other fields such as file size.

```
: Output full path but sort by file name.
XDir * /Form=*f /Sort=*F
```

/Execute

Executes the resulting output as a batch file.

```
: Normal: Spcify uppercase extensions for all text files.
*.txt
/Form=REN `*f` `*D*P*1.TXT`
/Execute
```

You can recursively execute.

```
: Recursive: For each folder called Debug*, delete all .obj files.
: Note the use of ** and `` to achieve * and " in the inner call.
```

```
XDir Debug* /NoFiles /Dirs "/Form=XDir *D*P**.obj `/Form=DEL ``**f``` /Execute"
         /Execute
```

/?

Prints some simple help information. If this parameter is specified, all others are ignored.

Globals Parameters

These global parameters are provided for use within in any parameter or within the form. They refer to the state of the PC at XDir launch time. They do not contain any information about any particular file. To access these parameters, enclose them in braces {}.

/Today=

Today's date in text form.
Format: dd/mm/ccyy

/Now=

Launch time.
Format: hh:mm:ss

/WDay=

Numeric day of week (Sunday = 0)
Format: One digit between 0 and 6.

/DayName=

Name of the day of the week.
Format: One of: SUN, MON, TUE, WED, THU, FRI, SAT.

/YDay=

Day number of year (1 – 365/6)
Format: 3 digits.

/ZToday=

Number of days since Sunday 1st Jan 1984.
Format: 5 digits.

/Yesterday=

Yesterdays date.
Format: Same as /Today=

/Here=

The folder we were launched from.
Format: Text

/Home=

The folder where XDir resides.
Format: Text

/Tmp=

Temp folder defined by environment variable TMP or TEMP..
Format: Text

Form Fields

These parameters are only available within the /Form= and /Sort= parameters but all global and command-line parameters are also available. They refer to details about the file. To access these parameters, enclose them in braces '{}' or use the short form if there is one available.
Details about the file are included using any of the special codes or parameter values listed later.

Short-form parameters.

Most parameters are available in short form for easier typing. The short forms are case sensitive and must be preceded by a '*'. To insert a '*', use '**'. If an invalid short form is specified it is treated as normal text.

Formatting short-form parameters.

To format short-form parameters, place the format in square brackets immediately after the form character. For example *d[78904512] will generate a date in the form "ccyymmdd".

Special characters in forms.

Normal characters in the form are repeated in the output with the exception of:

```
'¬'      Becomes <Esc> character.
'`'      Becomes quotes character. `` Becomes `.
'\xx'    Becomes the character specified by the hex code 'xx'.
'\t'     Becomes <Tab> character.
'\n'     Becomes <Return> character.
'\\'     Becomes '\' character.
```

Selective output

If the first character of the form is a '?' it indicates that the following bracketed expression should be evaluated to determine if the form should be output. Form expressions are defined as follows:

```
(<string1>'<comparator>'<string2>)
```

or

```
(#<numeric expression1>'<comparator>'< numeric expression2>)
```

Comparators can be one of:

```
'L'      Strictly less than.
'G'      Strictly greater than.
'E'      Equal.
'l'      Less than or equal.
'g'      Greater than or equal.
'e'      Not equal.
```

Numeric expressions can contain operators '+-*/%' with the obvious meanings and are evaluated strictly from left to right before the comparison. Brackets '()' are not allowed in numerical expressions.

```
: List only files more than 7 days old.
/Form=?(#{*z'L'{/ZToday=}-7)*D*P*F*E
```

~ and ^

File name forms (8.3 and long) can be switched between using a ~ to select 8.3 and ^ to return to long form.

The field parameters

The short form is shown to the right of the parameter name.

/Drive= ****D***

Drive letter with colon.

/Dir= ****P***

Full path of enclosing folder if object is a file or folder if object is a folder, with terminating '\'.

/Name= ****F***

File name without extension.

/Ext= ****E***

File extension including leading '.'.

/SDir= *P (when short names requested)

Like /Dir= but in in 8.3 form.

/SName= *F (when short names requested)

Like /Name= but in in 8.3 form.

/SExt= *E (when short names requested)

Like /Ext= but in in 8.3 form.

****D*P*F*E *f***

Special short form to include the full spec of the file.

/Date= *d

Last modification date.

/Time= *t

Last modification time.

/DateCr= *O

Creation date. (O Stands for Original).

/TimeCr= *o

Creation time. (O Stands for Original).

/DateAc= *X

Last access date.

/TimeAc= *x

Last access time.

/AttArc= *aa

Archive attribute. ‘1’ if on, ‘0’ if off.

/AttRo= *ar

Read-only attribute. ‘1’ if on, ‘0’ if off.

/AttHid= *ah

Hidden attribute. ‘1’ if on, ‘0’ if off.

/AttSys= *as

System attribute. ‘1’ if on, ‘0’ if off.

/AttDir= *ad

Directory attribute. ‘1’ if on, ‘0’ if off.

/Size= *s

Size of file or total size of all files in directory matching mask.

/Count= *q

Number of files in directory matching mask or 0 if file.

/TSize= *S

Size of file or total size of all files in directory and subdirectories matching mask.

/TCount= *Q

Number of files in directory and subdirectories matching mask or 0 if file.

/CRC= *C

8 character CRC32 Checksum of file. Algorithm matches that of WinZip. Useful for detecting duplicate files.

/Sha1= *H

Twenty character SHA1 digest of file contents. Useful for detecting duplicate files.

/Seq= *+

Count of files found so-far. 8 Digits. Useful as unique number for copying or renaming.

/ZDate= *z

Zero based day number of last modification date. Days since Sunday 1st Jan 1984.

/ZDateCr=

Zero based day number of last modification date. Days since Sunday 1st Jan 1984.

/ZDateAc=

Zero based day number of last access date. Days since Sunday 1st Jan 1984.

/YDay= *y

Day number of year. 1 = January 1st.

/Match1= to /Match9= *1…*9

The parts of the file name matched by wild characters '*' and '?'.

/Path0= to /Path9= *p0…*p9

The parts of the path starting from the right.

/Path-1= to /Path-9= *p-0…*p-9

The parts of the path starting from the left.

/Lc, /Uc and /Nc

Requests the following characters to be converted to lowercase, uppercase or no change. Useful for renaming.

Some of my favourite uses.

Show all folders with the total size of its contents.

Excellent for tracking down your most disk-hungry folders.

```
XDir /NoFiles /Dirs /S /Form=*D*P*F*E\t*S>XDir.txt
```

Rename a folder to today's date.

Useful for archiving of downloaded files. Renames a folder called 'Folder' to todays date with the form ccyymmdd.

```
XDir Folder /NoFiles /Dirs /Execute "/Form=REN *D*P*F*E {/Today=[78904512]}
```

Delete all files older than 7 days.

Useful for cleaning up archives.

```
XDir /Execute "/Form=?(#{/ZDate=}'L'{/ZToday=}-7)DEL *D*P*F*E"
```

Delete empty folders.

Useful for removing old archive folders after cleanup.

```
XDir /NoFiles /Dirs /Execute "/Form=?({/TCount=}'E'0)DEL *D*P*F*E"
```

Recent Enhancements

Version 1.3.0

- No more nags.

Version 1.2.8

- Force a '\' on the end of /Tmp= if its not there.
- /Quiet suppresses the 'Executing' message.

Version 1.2.7

- Added /Tmp= global and use of it for sort and execute temp files.

Version 1.2.6

- Fixed bug where /Sort also implied /S (again) and added '#' format for zero suppression.

Version 1.2.5

- Fixed bug where /Sort also implied /S.

Version 1.2.4

- Nothing noticeable.

Version 1.2.2

- Some more minor wild match issues resolved.
- No need to use '`\\*`', '`\*`' now works as expected.

Version 1.2.1

- Can now use short forms in expressions.

Version 1.2.0

- Sorting of the output. /Sort=
- Recursive execution now functions as expected.
- Removal of the /Unix parameter. Its always unix form now.

Version 1.1.2

- Allow '-' in format to indicate a range.
- Unix style is now always on.

Version 1.1.1

- Getting from file contents.
  Now uses `<>` instead of `[]` to avoid confusion with formats.

- Crash bug removed when using **??** in file mask.
- Multiple /Head= and /Tail=

Version 1.1.0

- Getting from file contents.
  You can now collect data from the file being processed.
- /Head= and /Tail=
  You can now add lines at the start and end of the run.
- /Execute
  You can now execute the output directly.
- *0-*9
  You can now access the parts of the file name that were matched by wild card characters.

Help me improve XDir!

E-mail me (XShareware@Yahoo.co.uk) a PayPal payment. Leave a message in my GuestBook at:
http://uk.geocities.com/xshareware
Get the latest version here:
http://uk.geocities.com/xshareware/Download/XDir.zip