

PowerTools_lib

COLLABORATORS

	<i>TITLE :</i> PowerTools_lib		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		November 28, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	PowerTools_lib	1
1.1	PowerTools_lib.doc	1
1.2	PowerTools.lib/--background--	1
1.3	PowerTools.lib/pt_BusyPointer	3
1.4	PowerTools.lib/pt_SimpleRequest	4
1.5	PowerTools.lib/pt_FileRequest	4
1.6	PowerTools.lib/pt_SplitPath	5
1.7	PowerTools.lib/pt_AllocProgress	6
1.8	PowerTools.lib/pt_UpdateProgress	7
1.9	PowerTools.lib/pt_FreeProgress	7
1.10	PowerTools.lib/pt_AllocListRequest	8
1.11	PowerTools.lib/pt_ListRequest	8
1.12	PowerTools.lib/pt_FreeListRequest	9

Chapter 1

PowerTools_lib

1.1 PowerTools_lib.doc

```
--background--  
pt_BusyPointer()  
pt_SimpleRequest()  
pt_FileRequest()  
pt_SplitPath()  
pt_AllocProgress()  
pt_UpdateProgress()  
pt_FreeProgress()  
pt_AllocListRequest()  
pt_ListRequest()  
pt_FreeListRequest()
```

1.2 PowerTools.lib/--background--

PowerTools_lib was created by Bart Vanhaeren.
©1993,1994 Quadra development.

The PowerTools package (lib, includes and docs) are public domain and freely distributable. Anyone who programs the Amiga may use the PowerTools code for commercial or PD software. A small note in the documentation of your software would be appreciated, though.

NOTE #1: THE POWERTOOLS.LIB FILE MAKES CALLS TO THE REQTOOLS.LIBRARY WHICH IS ©NICO FRANÇOIS. PLEASE NOTE THAT HIS DEMANDS AND RULES ACCORDING TO THE REQTOOLS PACKAGE ALSO STAY EFFECTIVE WHEN YOU USE IT THROUGH POWERTOOLS.LIB.

NOTE #2: POWERTOOLS.LIB REQUIRES OPERATING SYSTEM 2.04 OR HIGHER.

NOTE #3: NO WARRANTIES ARE IMPLIED OR EXPRESSED WITH REGARD TO THE FITNESS OR MERCHANTABILITY OF POWERTOOLS FOR ANY PARTICULAR PURPOSE. ALL RISKS AND DAMAGES, INCIDENTAL OR OTHERWISE, ARISING THROUGH THE USE OR MISUSE OF POWERTOOLS ARE ENTIRELY AT THE RESPONSIBILITY OF

THE USER.

What is PowerTools?

~~~~~

The PowerTools.lib is a standard Amiga linker-library that contains some handy and easy to use C-coded functions. The functions were developed and elaborated in a series of programs that I made. Because the functions increased the compilation time considerably, I thought it was a good idea to put them away in a library. This library will grow bigger as I have new and good functions to put into it. You can see it as a sort of amiga.lib.

Although most functions in this lib already exist in other (shared) libraries, the versions in my lib are easier to use, and very powerful too. In fact, this lib uses the existing code in other libraries but it also makes the necessary memory allocations and initializations. So, you don't have to bother anymore on opening libraries, closing them, declaring structures, filling them in, etc...

The functions in the lib are designed to make efficient use of the available libraries (e.g. PowerTools.lib uses Kickstart 3.0 features if available). Some functions allow you to prefer ReqTools routines without being dependant of ReqTools (if ReqTools calls fail, standard system routines will be used). This allows you to create very flexible and intelligent programs.

How to use PowerTools?

~~~~~

Currently, PowerTools can only be used within C-programming environments. If anyone knows how to make PowerTools available to other programming languages, please contact me.

Simply #include the PowerTools.h file (should be placed in INCLUDE:-directory) in your C-source. It contains all definitions and prototypes to use PowerTools.lib.

After that, you can call the PowerTools.lib functions like any other C-function. Just pass the right parameters and collect the (optional) return code, PowerTools does the rest!

After compilation, you must link your program object with the PowerTools.lib file. If you use the PT_USERT flag with the PowerTools functions, you must also link the reqtools.lib (together with PowerTools.lib) with your program. The reqtools development package is available from Nico François directly, or from various PD collections. The linker (from your C-development package) will attach all used PowerTools (and ReqTools) functions to your program. Now you can test your program and see if everything works as you expected.

PowerTools.lib opens all it's needed libraries itself, and closes them when they aren't needed anymore. This means you don't have to take care of opening and closing libraries for PowerTools, but it also means that you may *NOT* rely on the libraries PowerTools opened (because PowerTools may close them without any notice). If you use the same libraries as PowerTools, you should open (and close) them yourself, for private use. The only libraries PowerTools expects to be open are the exec.library and

the dos.library. Both of them should have been opened by the startup-code anyway.

Thanks to:

~~~~~

- Frank Maesen for  $\beta$ -testing and being the first PowerTools user.
- Nico François for the great ReqTools.
- Commodore-Amiga for creating my Amiga 2500/020.

Comments, suggestions, bug reports, questions and donations:

~~~~~

Quadra Development,
t.a.v. Bart Vanhaeren,

Weg naar Zwartberg 248
B-3660 Opglabbeek (BELGIUM)

1.3 PowerTools.lib/pt_BusyPointer

NAME

pt_BusyPointer -- Intelligent busypointer function (V1)

SYNOPSIS

pt_BusyPointer (window,state);

void pt_BusyPointer (struct Window *,BOOL);

DESCRIPTION

First checks if running under OS 3.0. If intuition.library V39++ could be opened, pt_BusyPointer uses the Workbench busypointer (user defined). If OS 3.0 check is negative, this function checks for the presents of reqtools.library (any version) and uses its rtSetWaitPointer() function. If this doesn't work, the function returns without changing the pointer. Window input is blocked by opening an invisible requester on it (V2).

INPUTS

window - a pointer to the window that should be (un)blocked for mouse input.
state - TRUE to block input, FALSE to unblock.

NOTE

Don't forget to call this function twice: the first time to block input and, very important, the second time to UNBLOCK the input.

BUGS

No busypointer when no intuition.library V39 or reqtools.library available. No info on this.

SEE ALSO

intuition.library/SetWindowPointerA() , reqtools.library/rtSetWaitPointer()

1.4 PowerTools.lib/pt_SimpleRequest

NAME

pt_SimpleRequest -- Very easy to use system requester (V1)

SYNOPSIS

```
num = pt_SimpleRequest (window,title,textformat,gadgetformat,flags);
```

```
LONG pt_SimpleRequest(struct Window *,STRPTR,STRPTR,STRPTR,WORD);
```

DESCRIPTION

This function allocates, interprets and frees everything necessary to display a system-requester. If you set the PT_USERT flag, pt_SimpleRequest() opens a ReqTools-requester. Otherwise, intuition will be used. This function supports multigadget requesters (any number of gadgets) and a C-style formatted text string (without arguments).

INPUTS

window - a pointer to the window the requester should appear on. If you pass NULL, the requester opens on the default public screen.
title - pointer to a title string.
textformat - C-style formatted string with the body text
(e.g.. "Hello World!\nThis is an example...").
gadgetformat - string containing all gadgetlabels, each label separated by a '|' -symbol (e.g. "OK|Maby|Cancel").
flags - Currently supported flags:

PT_USERT: Set if you want reqtools.library to be used.
If this fails, the function will fall back
on intuition (which should work always...)

RESULT

Number of gadget that was selected. Note the strange numbering:
From left to right, the gadgets return number 1, 2, etc.
The utmost right gadget (the "Cancel" gadget) returns number 0.

NOTE

This function calls pt_BusyPointer() internally if window ~= NULL.

BUGS

SEE ALSO

intuition.library/EasyRequestArgs() , reqtools.library/rteZRequestA()

1.5 PowerTools.lib/pt_FileRequest

NAME

pt_FileRequest -- Very easy to use file requester (V1)

SYNOPSIS

```
ret = pt_FileRequest (window,title,pathbuffer,pattern,flags);
```

```
BOOL pt_FileRequest(struct Window *,STRPTR,UBYTE *,STRPTR,WORD);
```

DESCRIPTION

This function allocates, interprets and frees everything necessary to display a filerequester. If you set flags to PT_USERT , pt_FileRequest opens a ReqTools-requester. Otherwise, an Asl-requester will be used.

INPUTS

window - a pointer to the window the requester should appear on. If you pass NULL, the requester opens on the default public screen.
 title - pointer to a title string.
 pathbuffer - pointer to a stringbuffer which should be at least 108 bytes long
 pattern - pointer to a string that contains a pattern to be used in the file-requester. NULL if no pattern required.
 flags - Currently supported flags:

PT_USERT: Set if you want reqtools.library to be used.
 If this fails, the function will fall back on asl (hope this works !).

PT_SAVEMODE: Set if the requester is used for a save or write operation (strongly advised!).

RESULT

TRUE if user selected OK (string in pathbuffer is valid), FALSE if user canceled requester (do not use pathbuffer contents !).

NOTE

The allocation and length of the buffers are completely your responsibility ! Buffers of less the 108 bytes may lead to serious memory trashing (and machine crashing)!!

BUGS

No filerequester when reqtools.library and asl.library are not available. No info on this.

SEE ALSO

asl.library/AslRequest() , reqtools.library/rtFileRequestA()

1.6 PowerTools.lib/pt_SplitPath

NAME

pt_SplitPath -- Break a path into a dirname and a filename (V1)

SYNOPSIS

```
pt_SplitPath (path,directorybuffer,filenamebuffer);
```

```
void pt_Splitpath(STRPTR,UBYTE *,UBYTE *);
```

DESCRIPTION

This function splits a given path into a directoryname and a filename. The two name-strings will be copied into the directory and filename buffers.

INPUTS

path - pointer to a legal AmigaDOS path string. (e.g. the one that

pt_FileRequest creates in your pathbuffer)
directorybuffer - pointer to a directory buffer which should be at
least 108 bytes long.
filenamebuffer - pointer to a filename buffer which should be at
least 108 bytes long.

NOTE

The allocation and length of the buffers are completely your responsibility ! Buffers of less the 108 bytes may lead to serious memory trashing (and machine crashing)!!

BUGS

SEE ALSO

dos.library/FilePart() , dos.library/PathPart()

1.7 PowerTools.lib/pt_AllocProgress

NAME

pt_AllocProgress -- Allocate, initialize and render progress indicator (V1)

SYNOPSIS

```
progress = pt_AllocProgress (window,xcoord,ycoord,width,height,max,flags);
```

```
struct pt_Progress *  
pt_AllocProgress(struct Window *,UWORD,UWORD,UWORD,UWORD,ULONG,UWORD);
```

DESCRIPTION

This function allocates an pt_Progress structure, fills it with the init-values and renders a progress indicator in the window according to the guidelines in the Amiga User Interface Style Guide (Commodore-Amiga). It returns a pointer to the pt_Progress structure for the indicator.

INPUTS

window - a pointer to the window the indicator should render in.
xcoord - X-coordinate of the indicator box. Note that before the box a "0% " string will be rendered (leave enough space !).
ycoord - Y-coordinate of the indicator box.
width - width of the indicator box. Note that after the box a " 100%" string will be rendered (leave enough space !).
height - height of the indicator box.
max - maximum level for indicator (= total = 100%), e.g. the total size of a file, the total estimated time for a job. This value will be used to scale the progress indicator level while updating.
flags - not used, for future enhancements. Currently always NULL.

RESULT

A pointer to an initialized pt_Progress structure. This structure contains everything the pt_UpdateProgress() function must know to adjust the level in the indicator. All fields in this structure are private (and subject of change in future) and may only be changed by the specific PowerTools functions.

BUGS

Rendering is based on topaz.8 font and this font is considered as the

default screen font (and this is not always the case under Release 2++).

SEE ALSO

`pt_UpdateProgress()`, `pt_FreeProgress()`

1.8 PowerTools.lib/pt_UpdateProgress

NAME

`pt_UpdateProgress` -- Display progress (update) in a progress indicator (V1)

SYNOPSIS

`pt_UpdateProgress (progress,progressvalue);`

`void pt_UpdateProgress(struct pt_Progress *,LONG);`

DESCRIPTION

Use this function to keep the progress indicator up-to-date. According to the progress value you pass, `pt_UpdateProgress` will render the portion of the job that has been processed (in % scale). You can call this function at any time, as much as you want. I advise you, though, not to call this function after every time a tiny bit of the job has been completed. The update routines performs a few (heavy) float calculations which require some processor time. I suppose you don't want your program calculating on the updating all the time, instead of working on your job. A periodic call to this function during the processing of the time consuming job should not slow down the performing of your program, however.

INPUTS

`progress` - pointer to the `pt_Progress` structure of the indicator.
 `pt_UpdateProgress()` will update some fields of the struct.
`progressvalue` - the absolute amount of the job already finished. This value must always increase (with every call to this function) until it finally equals to the max value passed in the `pt_AllocProgress()` function.

SEE ALSO

`pt_AllocProgress()`, `pt_FreeProgress()`

1.9 PowerTools.lib/pt_FreeProgress

NAME

`pt_FreeProgress` -- Free resources `pt_AllocProgress` allocated (V1)

SYNOPSIS

`pt_FreeProgress (progress,flags);`

`void pt_FreeProgress(struct pt_Progress *,UWORD);`

DESCRIPTION

When you do not need the progress indicator anymore, you MUST call this function to free all the memory `pt_AllocProgress` allocated.

INPUTS

progress - pointer to the pt_Progress structure of the indicator.
flags - not used, for future enhancements. Currently always NULL.

SEE ALSO

pt_AllocProgress(), pt_UpdateProgress()

1.10 PowerTools.lib/pt_AllocListRequest

NAME

pt_AllocListRequest -- Allocate and initialize ListRequester (V2)

SYNOPSIS

```
listreq = pt_AllocListRequest(window,font,flags);

struct pt_ListRequest *
    pt_AllocListRequest(struct Window *,struct TextAttr *,UWORD);
```

DESCRIPTION

Allocates and initializes a pt_ListRequest data structure for you.

INPUTS

window - pointer to the parent window (you must supply this !).
font - font to render ListRequest (NULL is not safe!).
flags - not used, for future enhancements. Currently always NULL.

RESULT

A pointer to an initialized pt_ListRequest structure. This structure must be passed to the pt_ListRequest() function. It is safe to read/write the fields in the struct.

SEE ALSO

pt_ListRequest(), pt_FreeListRequest()

1.11 PowerTools.lib/pt_ListRequest

NAME

pt_ListRequest -- Get input from user via ListRequester (V2)

SYNOPSIS

```
item = pt_ListRequest(listreq,title,list,flags);

WORD pt_ListRequest(struct pt_ListRequest *,STRPTR,struct MinList *,UWORD);
```

DESCRIPTION

Opens a ListRequester presenting a list of possible choices to the user. This function returns when the users makes a choice (returncode >= 0) or cancels it (returncode = -1). pt_ListRequest() uses pt_BusyPointer() to block all input for the parent Window. It also uses the parent Window to find the Screen to open on. The ListRequester is sizeable and font-

sensitive.

INPUTS

listreq - pointer to an initialized pt_ListRequest structure.
title - title string.
list - pointer to a valid MinList structure to display in the requester.
flags - not used, for future enhancements. Currently always NULL.

RESULT

Ordinal number of selected item (starting from zero) or -1 if requester failed or canceled by user. The selected item (struct node *) can be found in the listreq->lr_node field.

SEE ALSO

pt_AllocListRequest(), pt_FreeListRequest()

1.12 PowerTools.lib/pt_FreeListRequest

NAME

pt_FreeListRequest -- Free resources pt_AllocListRequest allocated (V2)

SYNOPSIS

```
pt_FreeListRequest (listreq);  
  
void pt_FreeListRequest(struct pt_ListRequest *);
```

DESCRIPTION

When you do not need the ListRequester anymore, you MUST call this function to free all the memory pt_AllocListRequest allocated.

INPUTS

listreq - pointer to the pt_ListRequest structure.

SEE ALSO

pt_AllocListRequest(), pt_ListRequest()
