ProWesS documentation

PROGS, Professional & Graphical Software
Dr. Frans Hemerijckxlaan 13 /1
2650 Edegem
BELGIUM
tel : +32 (0)3/ 457 84 88 fax : +32 (0)3/ 458 62 07 e-mail : joachim@club.innet.be
www : http://www.club.innet.be/~year2827

Utility software

ProWesS reader

The program to read the manuals etc. which is also used as help system for ProWesS applications.

ProWesS loader

ProWesS documentation

PROGS, Professional & Graphical Software
Dr. Frans Hemerijckxlaan 13 /1
2650 Edegem
BELGIUM
tel : +32 (0)3/ 457 84 88 fax : +32 (0)3/ 458 62 07 e-mail : joachim@club.innet.be
www : http://www.club.innet.be/~year2827

Utility software

ProWesS reader

The program to read the manuals etc. which is also used as help system

for ProWesS applications.

ProWesS loader
: The program which, together with some smaller utilities, allows loading of extra applications, and runtime configuration management. This makes it possible to add or start application without resetting the computer.

ProWesS calculator
: A simple calculator, which also allows conversion between decimal, hexadecimal and binary numbers.

procon
: A config program which allows you to configure programs which contain level 1 and/or level 2 config blocks.

PFconfig
: This utility helps you to configure all aspects of PROforma. You can add drivers and fonts and other options. The program also helps you to configure the imageable area of the printer driver, so that it matches your model.

PWconfig
: A special program to modify the ProWesS and PROforma config files. All changes which are made to the file can be saved, and they are also passed to ProWesS or PROforma respectively, to make the effects visible immediately.

ProWesS introduction

What is ProWesS

ProWesS is short for '*PROGS Windowing System*' and is (as the name suggests) a new window manager. However, the ProWesS package contains a lot more than just ProWesS (in fact ProWesS is just the most important part of a group of programming libraries).

However, the ProWesS package also contains some utility programs, like the ProWesS reader.

ProWesS manual

This is the general ProWesS manual, which explains how to use ProWesS and the utility programs which are part of the ProWesS package. It does *not* explain how to write programs which use the system extensions which are part of ProWesS. These manuals are available in public domain (electronically and via PD suppliers).

We (and everybody who uses these manuals) would like it very much if you could send us any comments about this manual, like

- omissions
- inaccuracies or mistakes
- typing and/or spelling mistakes
- making this manual into better English

- anything else (positive comments are also always appreciated)

At the bottom of each page is mentioned when the HTML document was last modified. I will try to keep this date correct, however it is only meant to indicate changes in the information provided, I will not change that date when correcting spelling mistakes or HTML errors.

Disclaimer & Copyright

is copyrighted by Wolfgang Lenerz.

- the *PWconfig* program allows you to modify (both permanently and temporarily) the ProWesS and PROforma configuration. This program is copyrighted by Wolfgang Lenerz.
- the *procon* program allows you to configure a file which contains standard level 1 or 2 config blocks. This program is copyrighted by Wolfgang Lenerz.
- The IBM Courier font which is distributed as part of ProWesS, is produced and copyrighted by IBM Corporation.

  IBM Courier - Copyright © IBM Corporation 1990, 1991

  You are hereby granted permission under the terms of the IBM/MIT X Consortium Courier Typefont agreement to execute, reproduce, distribute, display, market, sell and otherwise transfer copies of the IBM Courier font to third parties.

  The font is provided "AS IS" without charge. NO WARRANTIES OR INDEMNIFICATION ARE GIVEN, WHETHER EXPRESS OR IMPLIED INCLUDING, BUT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

- The Bitstream Charter font which is distributed as part of ProWesS, is produced and copyrighted by Bitstream Inc.

  © Copyright 1989-1992, Bitstream Inc., Cambridge, MA.

  You are hereby granted permission under all Bitstream propriety rights to use, copy, modify, sublicense, sell, and redistribute the 4 Bitstream Charter (R) Type 1 outline fonts for any purpose and without restriction; provided, that this notice is left intact on all copies of such fonts and that Bitstream's trademark is acknowledged as shown below on all copies of the 4 Charter Type 1 fonts.

  BITSTREAM CHARTER is a registered trademark of Bitstream Inc.

- The Utopia font which is distributed as part of ProWesS, is produced

and copyrighted by Adobe Systems Incorporated. Utopia is a registered trademark of Adobe Systems Incorporated.

Permission to use, reproduce, display and distribute the listed typefaces is hereby granted, provided that the Adobe Copyright notice appears in all whole and partial copies of the software and that the following trademark symbol and attribution appear in all unmodified copies of the software:

Copyright © 1989 Adobe Systems Incorporated
Utopia (R)
Utopia is a registered trademark of Adobe Systems Incorporated

The Adobe typefaces (Type 1 font program, bitmaps and Adobe Font Metric files) donated are : Utopia Regular, Utopia Italic, Utopia Bold, Utopia Bold Italic.

- The ProWesS reader contains an SGML parser which originates from the W3C library.

  Copyright 1995 by: Massachusetts Institute of Technology (MIT), CERN

  This W3C software is being provided by the copyright holders under the following license. By obtaining, using and/or copying this software, you agree that you have read, understood, and will comply with the following terms and conditions:

  Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee or royalty is hereby granted, provided that the full text of this NOTICE appears on ALL copies of the software and documentation or portions thereof, including modifications, that you make.

  THIS SOFTWARE IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES OF MERCHANTABILITY OR

FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. COPYRIGHT HOLDERS WILL BEAR NO LIABILITY FOR ANY USE OF THIS SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

(webmaster@w3.org, May 1995)

Please note that this copyright message only applies to the SGML parser which is part of the ProWesS reader program. The rest of the program is copyright 1996 by Joachim Van der Auwera, for PROGS, Professional & Graphical Software.

You are not free to distribute the ProWesS reader without written permission from PROGS. However, you can get the modified SGML parser, as it is part of the ProWesS reader. This means the SGML_c, HTChunk_c and HTMLdtd_c files and associated header files, HTStruct_h and HTStream_h.

- the *Qlib_sys* file contains the *Q_Liberator Runtimes* and is copyrighted by Liberation Software. These runtimes are used in some of the utility programs (e.g. procon & PWConfig)

All the code which is copyrighted by PROGS is written by Joachim Van der Auwera, except the ProWesS calculator, which is written by Nathan Van der Auwera.

The *Complete* font is copyrighted by PROGS with all rights reserved. It was designed and written by Nathan Van der Auwera.

Although much care is taken in the development of the ProWesS package and manual, in no circumstances will PROGS, Professional & Graphical

Software, be liable for any direct, indirect or consequential damage or loss arising out of the use or inability to use any part of the PROforma software or documentation.

This said, it goes without saying that PROGS will continue to develop this manual and software. Therefore, we would appreciate any comments about our software and manual. As you may know, we are only human, we can do no more than our best to provide you with the best quality.

Present, Past and Future

Past

There has been a *pre-release* verssion of ProWesS. This was mainly intended to allow interested people to start using all the (new) libraries and get acquainted with the system. Between the first pre-release (January 28, 1996) and the first release version (September 7, 1996) many things have still changed, but all for the better. The system has been improved, some utilities have been added, installation of programs is now possible, and the performance of many parts has been improved.

Present

This is the first release version of ProWesS. Although the package can still be improved in many ways, we feel that, after two years of work, it is time to get a user base and allow you to make use of the work we have done. ProWesS now includes

- The full set of libraries.
- The ProWesS reader, for reading manuals and help files.
- The ProWesS loader, for starting programs without resetting (and also needed for automatic program installation).
- Some simple ProWesS program, like a calculator.
- Installation software.

Future

However, some things can still be improved. For example, we would like to get some work done in the following areas (any help is always appreciated) :

- Some extra simple programs, e.g. to select the default printer driver, to preview fonts etc.
- Maintenance software, to control the current ProWesS system, possibly adding or removing parts (e.g. printer drivers, installed fonts) to your ProWesS installation.
- More picture and printer drivers for PROforma.
- ProWesS can be improved in some areas, like to make menus adjust their own size depending on the number of items in it.
- Some extra types would be useful in ProWesS, for example support for a text editing window (kind of a multi-line edline).
- ...

Installation

ProWesS can automatically be installed (as is also the case for all ProWesS applications). When your computer is booted with the ProWesS master disk, you immediately get the option to install it. To update your installation, you should indicate the "*install software*" item in the utilities button.

PROGS, Professional & Graphical Software
last edited September 5, 1996

ProWesS in pieces

ProWesS is a *software support package*, which basically means that is doesn't do very much by itself, but is essential to make some applications work on your computer. The package mainly consists of a group of system extensions (mostly programming libraries), which have to be present for some programs (ProWesS applications) to work.

Apart from that, ProWesS also contains some programs which actually do something. The most important one (may be even the reason for buying ProWesS) if the ProWesS reader which is used to read HTML documents like the ProWesS manuals.

Another important group of programs are used for configuration and system management, and allow you to start programs without rebooting (see ProWesS loader). A program for automatic installation of applications will also be provided in future.

The system extensions which are part of ProWesS are :

- Rationale
- ProWesS
- PROforma
- syslib
- DLL Manager
- DATAdesign engine
- other extensions

Rationale

While we all know what a wonderful computer we have, and most of us don't even consider using anything else (I know I don't), it is not all that practical for writing large applications. As a result, the release of new applications are rare. To try to remedy this problem, we have started developping a new set of

programming libraries. However, as we did not want to loose the aspects which are rather typical for our OS, we have had to make a clean break from already existing libraries (like standard c libraries as part of c68). So ProWesS applications can still be re-entrant and are quite small.

ProWesS

ProWesS is built with the following goals :

- *Ease of programming.* The creation of the user interface of a program is an important task. However, the efforts should go towards making the interface powerful and easy to use, and towards making it do what it should.
- *Configurable.* It should be easy to change the parameters which determine what a program looks like. If you prefer small text for more information, or large text for better readability, that should be easily configured. In essence, many parameters can be determined about how things are displayed. It should be possible to change these.
- *Consistency.* A window manager can be used by many programs. It is preferable if all these programs are somewhat consistent. So instead of configuring each program individualy, it would be better to have the general parameters globally configurable. This way each application automatically fits in with the rest, and the programmer is not burdened with it.
- *Fast prototyping.* ProWesS is designed to allow windows to be created with a limited amount of work. Some extra effort may be required to make it look properly, and definitely to make the scaling work as intended. So ProWesS allows you to first concentrate on making a GUI application that works, and worry about the details later.
- *Screen independence and PROforma support.* ProWesS is a general framework which has been designed specifically to allow the use of PROforma for all drawing. This way all the text on your screen can be drawn with the font and size of your choice. Because ProWesS uses PROforma for the drawing, it is possible to have windows which are larger than the screen, and also screen independence. When using a high resolution monitor, the fonts will still be as big as on a screen with less resolution.

- *Re-entrant code*. It should be possible to write re-entrant programs using ProWesS. Re-entrant code can be executed many times with only one copy in memory. This means that a mechanism for accessing global data has to be provided - without using global variables. This is particularly important because ProWesS is event driven which means that the window manager calls the routines (and the programmer can not determine the prototype).

ProWesS is strictly event driven. The programmer has to describe what the window should look like. The control is then passed to ProWesS which will display the window and wait for events. The events can be handled by routines which are provided by the programmer. These routines can change the behaviour of the window, add or remove something from the window or do something (like copy a file). When the event handler terminates, the window is updated. When the control is passed back to the calling application (because the window is exited (or broken down in ProWesS terminology)), the window is removed from the screen.

PROforma

The OS has never given a lot of support for graphics. This has changed when PROforma was released. We therefore no longer support most of the graphics in syslib, but recommend the use of PROforma when graphics are necessary.

PROforma is short for '*PROGS Font & Raster Manager*', and it does exactly what this name suggest. It is a library of routines to manage and display vector graphics and fonts on (raster) devices like screens and printers.

The availability of a separate program to manage graphics and fonts has several advantages. It allows application developers to create output of equal quality (resolution permitting) on several devices, and they can share resources. In short this means that the PROforma library only has to be loaded once, independent of the number of applications which use it. Also fonts only have to be loaded once, and can be shared between applications.

PROforma was originally developped as the graphics library for LINEdesign. That does not mean that this is the only kind of application for which

PROforma is of use. PROforma is also perfectly suitable for desktop publishers, word processors, business graphics and all applications which want high quality output (which must be just about every application except compilers and games). Actually, even at the time of writing there are things which are possible with PROforma but can't be accessed through LINEdesign.

More recently, PROforma has been redesigned to a great extent, to make it even more future-proof, easier to extend (both internally, and by writing drivers). There have been some changes to make it easier to write a window manager (for ProWesS) and complete support of colour has been added.

As a library, PROforma has the form of dynamic link library (DLL).

syslib

syslib is a set of low level library routines, mainly to provide an interface with the operating system (although some user code is included as well). The library is specifically written to be incompatible with the standard C library so that they can be mixed if necessary (although a lot of it is quite similar to what is already available).

The standard C library is specifically written for unix® machines and presents a few problems when compared to syslib. In fact it is difficult to implement the standard libraries completely. This requires the support of signals, which is some kind of user interrupt of a program. However much more annoying is the limited error handling. Normally errors are reported by returning an out of the ordinary value to functions, and storing what the problem was in `_errno`, a global variable.

Unfortunately, global variables make programs modify their own code, in which case the code is not re-entrant. As the standard libraries always use these variables, most C programs are not re-entrant. However, if no global variables are used then there are no problems. syslib specifically does not require the use of global variables. When library functions need to read or modify global variables, proper DLL linking also becomes difficult to provide.

Consistency is also a strong point of syslib. There is a consistent naming scheme, only one memory model, only one type of strings etc.

syslib is also intended to make programming safe. No unsafe constructs are supported by syslib, or when possible, a safe way of doing things is provided. For example, many aspects of QDOS are only accessible because of direct access to the system data structures. However, this can cause dangerous modifications and allows you to do some things which are not "clean". syslib specifically does not support such operations.

An example of an important part of the library is the support for external modules. This allows code to be written as a separate part of the program, which can be loaded on demand. This can be useful because it allows pieces of a program to be replaced by others, or extra pieces to be loaded. This is for example very useful for printer drivers. Many drivers can be added when necessary (as indeed is the case in PROforma).

DLL Manager

C programs are typically quite large (especially when compared to native assembler). There are two main reasons for this. The least important factor is the translation from c to assembler. This translation produces code which is less optimal than native assembler (from good programmers) and can therefore be slightly larger. However this does not make a significant speed difference. Most of the time, a program is waiting for the user to tell it what to do anyway. For those pieces of code where speed does matter (typically less than 1% of a program) you could still use assembler anyway.

A much more important factor are the libraries which are included in the program. These libraries can make up a large part of the final program. As such, this is not a problem, but the same routines are linked with many programs, thus waisting memory when multitasking. Thus we wrote the *Dynamic Link Library* Manager. When a program uses the DLL Manager, it can link to routines which are separately loaded. In fact, a dynamic link library is just a kind of thing. However, it is used in such a way that the calls to the thing code are a lot more efficient than when using the extension thing mechanism.

Note that the DLL Manager does not load extensions, it only links to them. The direct loading of extensions is in general not possible without assuming that there is a harddisk with the extensions. Therefore the extensions have to be loaded in advance, and an error will be reported if a dynamic link can not be resolved. (In fact, ProWesS applications normally use a loader program which will test whether the needed extensions are available and if necessary load them - this is some kind of boot facility and only works in a controlled environment).

The DLL Manager also does prepares the system for using syslib (it makes sure that enough supervisor stack is available), and allows you to load all programs which use it as resident extensions. The programs will then be linked as executable things.

DATAdesign engine

The DATAdesign engine is a rather powerful, multi-user, free-form database management system. It allows DATAdesign files to be shared (used at the same time) by several programs (with proper record locking). The fact that it is *free-form* means that the maximum length of a field doesn't have to be specified in advance, but can vary from record to record. This allows you to store an entire text in a record.

Other extensions

Thing System

This is one of the most important extensions which exist. It allows a general (named) access mechanism to arbitrary pieces of memory. It is for example used by the DLL Manager for finding the DLL's. This extension was written by Qjump.

Pointer Interface

This is one of the basic system extensions, which allows the use of a *pointer*, the arrow or shape which is used to indicate things. It also does other things

like making sure that the window from one job is not overwritten by another job, and taking care of (re)displaying a window when switching jobs. The pointer interface was developped by Qjump.

Scrap Extensions

The Scrap extensions (which are produced and written by Jochen Merz software), is a buffer to pass data (usually text) between applications.

Hotkey System II

The *Hotkey System II* extension allows the user to define the definition of combinations of <*ALT*> with another keypress. Amongst the user defined combinations, it handles *last line recall* and the *Stuffer Buffer* (which is a one line scrap which is often used to pass filenames between applications). The hotkey system is developped by Qjump.

Global Variables

Global Variables are usually referred to in (DOS® and Unix®) literature as *Environment Variables*. It is a global mechanism to assign a string to a name, and make that globally available (in our case without a shell). This is very important to make automatic installation of software possible.

Global Variables are often used to indicate a directory, and can be used to start a directory name (possibly in a path).

Using ProWesS

ProWesS is intended to make things rather easy to do, so there is not a lot you have to know before you can fully use it.

You only have to know two little things about the windows in general, and you can do anything with a little experimentation.

The actual functionality of the items inside the window depends on the type of the items which are displayed. Not too much help is necessary here as most things are rather straightforward, but some things are explained in more detail.

General ProWesS usage

Scaleborder usage

A ProWesS window always contains a *scaleborder*, which can be used to move and scale the window. The pointer will change to a small diagonal bidirectional arrow when you are inside the scaleborder. When you *HIT* the scaleborder, you can move the window, a *DO* allows you to scale the window. Any keypress will terminate the moving or scaling. If your system is configured to do so, a preview of the window at the new position and/or size will be shown.

Note that not all windows are fully scaleable. A window may be limited to scale in one direction only, or even not at all. A window which can not be scaled will just let you move the window when scaling. When scaling, the opposite corner of the window will normally maintain its position.

All windows include a scaleborder at each side, *except* when the window is larger than can be visualised.

Larger than screen windows

In ProWesS it is quite possible and permitted for a window to be larger than can be shown. This can be noticed because the scaleborder is not visible in the orientations where the window is too large.

There are two possible reasons why a window may be too large.

- The window is larger than the screen.
- The window is larger than the *primary window* of the job. The pointer interface requires each window to have an *outline*, which is the maximum area that can be ocupied by that window. The *primary window* is the window which was first opened for a job (counting restarts when all windows are removed). In ProWesS this is the bottom window. All subwindows are limited to fall inside the primary, but this may be impossible.

If a window is too large to be visualised completely, it becomes scrollable, and the scaleborder is not displayed in the direction in which the window can be scrolled. To scroll the window, you should use <ALT> + <SHIFT> + <CTRL> + <cursor key>.

If you are using SMSQ/E on your system, you can also scroll windows that are larger than the screen by trying to move the pointer out of the screen (to touch the edge of the screen).

HIT, *DO* and keypresses

In ProWesS, all keypresses are case dependent, so the application programmer can assign different actions to upper and lowercase variants. However, if a keypress does not have an action attached to it, then the case is changed and another attempt is made to find an action for it.

A *HIT* is generated by pressing the left mouse key, and a *DO* by pressing the right mouse key. Usually, a *HIT* can also be generated by pressing <space>, and a *DO* by pressing <enter>. However, in some windows these keypresses are used in an input item.

In some objects *dragging* is possible. This is done by maintaining a *HIT* or *DO* somewhat longer than normal. This is usually only useful when the

pointer is also moved at the same time. A *dragging* operation starts when the mouse key is pressed down, and stops when the key is released.

Type specific help

Scroll Bar

A scroll object consists of arrows and possibly also a bar which indicates the approximate size and position of the visible part of the document. A *HIT* on the scroll arrow will usually scroll the window by a small amount (typically a "line"), a *DO* will scroll a larger part (typically a "line" less than the size of the scrollable area.

The reaction for a scroll bar is somewhat different. When the scroll bar is *HIT*, the approximate position which was indicated will be displayed. When you indicate the scroll bar with a *DO*, then the start or end will be displayed, whichever is closest to the position in the scroll bar which was indicated.

File Select window

The File Select Window allows you to indicate one or more files (depending on what it is used for). It mainly consists of a large scrollable area which displays the files in the current directory.

If only one file can be selected, a "filename" object is displayed. This object allows direct input of the filename. This object can also be selected by pressing <f>.

The window always has an "extensions" object. This object can be used to edit the list of extensions. Only files which end in one of the given extensions will be displayed in the window. Several extensions can be given by separating them by semicolons (';'). The selection of files can also be reversed by indicating the "Not" item. The "extensions" object can be selected by pressing <e>, the "Not" item can be indicated by pressing <n>.

There is also a "directory" object. If this is indicated by a *HIT* or <d>, then

you can edit the directory name. If you indicate the item with a *DO*, a directory select window is displayed. There is also a "<-" item, which allows you to move up in the directory tree (press <<>).

There is also a "Tree" object. When this is selected, then all the files in the subdirectories will also be displayed. This item can also be selected by pressing <t>.

When several files can be selected in the window, then there is also an "All" item which can be used to (de)select all the files. Press <a> for this item.

Directory Select window

The Directory Select Window can be used to select a directory. It displays the current directory, some subdirectories and some devices. The window may also display some default subdirectories which can be selected directly.

Which devices and default subdirectories are displayed can be configured in the `ProWesS_cfg` file.

The "directory" object indicates the current directory. It can be edited directly (press <d>), or it will be modified by indicating anything in the window. There is also a "<-" item, which allows you to move up in the directory tree (press <<>).

Edline object

It may be very useful to know that it is often possible to move between the edline objects while typing. This can be possible with any of <left cursor key>, <right cursor key>. <tab> or <shift tab>.

All of the standard keypresses can be used, including keypresses to move by (space separated) words (<shift left> and <shift right>), and to the start or end of the line (<alt left> and <alt right>). You can delete the entire line at once by pressing <ctrl down>, and the start or end of the line (<alt ctrl left> and <alt ctrl right>).

PROGS, Professional & Graphical Software
last edited March 27, 1996

Configuring ProWesS

Configuring ProWesS is quite straightforward. It basically boils down to editing the configuration files which are used by PROforma and ProWesS. In fact, there is even a special program which aids you in doing this.

Another important aspect is combining ProWesS with other programs which you want to have loaded in your system. Some guides are given below about integrating ProWesS in your existing boot files.

Please note that most applications can also be configured individually. This can be done using procon or another configuration program which can handle level 2 configuration blocks.

ProWesS by default displays some buttons to call some utility or application programs. This can also be customized to allow you to select the applications that you use often, remove some items, or add other buttons to call your favorite programs...

ProWesS

ProWesS is a highly configurable system, many parameters in the system can be changed. However, these can be situated either in the ProWesS program, or in the external types. Therefore, all configurable items have been grouped in a configuration file, called `ProWesS_cfg`, which is loaded when ProWesS is started (this file can be found in the `pws_pw` directory on the program disk).

Each line in the configuration file is interpreted as a configuration command. Empty lines are discarded as comments. All the other lines are divided into

two type : commands and definitions of configuration constants. The lines with a command have a fixed format : the first character is the actual command, the second character should be a space, and the rest of the line is the parameter. All lines which don't have a space as second character are considered as configuration constants.

The configuration commands currently supported by ProWesS are :

'%' and ';'
: the line is considered as comment and is discarded.

'S'
: set the searchpath for the following commands. The searchpath contains the directories which should be searched to open a file. The directories should be separated by a semicolon (';'). The directories are scanned from left to right. For each directory, the trailing underscore ('_') may be discarded.

'T'
: specify a type definition file which should be loaded. The file is searched on the current searchpath. A type is an external module (as supported by syslib), which contains the behaviours of the objects of that type. All types in ProWesS are external, ProWesS has no builtin types.

Lines which contain the definition of a configuration constant contain the name of the configuration constant, followed by the parameter, separated by one or more spaces or tabs. Each definition is passed to the system and each of the types for processing, which can thus modify their behaviour.

By convention, the name of the configuration constant starts with the type which is intended to process the result. However, all other types also see the definition and get a chance to modify their behaviour according. As the name of the configuration constant can not contain spaces, dashes ('-') are used to separate the words. The names are case dependant.

All the coordinates and widths are given in virtual screen coordinates (called points). These pretend a screen size if 720 by 540. There are some exceptions, `SYSTEM-BORDER-WIDTH`, `SYSTEM-SCALEBORDER-WIDTH`, `SYSTEM-SHADOW-RIGHT` and `SYSTEM-SHADOW-BOTTOM` have a parameter in pixels.

Colours are normally given as RGB colours. This means that the colour is split in *Red*, *Green* and *Blue* components in that order. The value "`100 100 100`" indicates white and "`0 0 0`" indicates black. The exceptions to this are `SYSTEM-BORDER-COLOUR` and `SYSTEM-SCALEBORDER-COLOUR` which need system colour values (0 for black, 2 for red, 4 for green and 7 for white).

The default configuration file which is on the program disk does not contain all the possible configuration constants which are accepted by the standard types. The possible definitions depend on the type of object.

Possible definition constants

- ProWesS system
- applic type
- edline/dedline types
- dirselect type
- separator type
- infotext type
- infostring type
- menu type
- title item type
- loose item
- scroll type
- item/itemp types
- label type
- listselect type

PROforma

Like ProWesS, PROforma also reads the initial configuration information from a special file called `PROforma_cfg`, which can be found in the `pws_pf` directory.

Each line in the configuration file is interpreted as a configuration command. Empty lines are discarded as comments. All the other lines are divided into two types : commands and definitions of configuration constants. The lines with a command have a fixed format : the first character is the actual

command, the second character should be a space, and the rest of the line is the parameter. All lines which don't have a space as second character are considered as configuration constants.

The configuration commands currently supported by PROforma are :

'%' and ';'
: the line is considered as comment and is discarded.

'S'
: the parameter is now the searchpath for fonts.

's'
: to set the searchpath for drivers.

'D'
: will load the given PROforma driver. It is not necessary to know what kind of driver it is. The names of PROforma driver files normally end in '`_pfd`'. The file will be searched on the current searchpath for drivers (cfr 's').

'M'
: allows you to specify the maximum amount of memory which can be used by PROforma as buffer to render a page in. If the amount given is negative, then that is the amount of memory which has to remain free (both in bytes).

'C'
: specify the size of the font cache. This consists of two numbers, the actual size of the fontcache, and the minimum number of different font/size combinations that can be in the cache (one more combination can be in the cache for each gstate). Each combination of font & size uses about 1.5kB of memory, so this number should not be too big, however, if you use a large fontcache, this number should also be increased.

'c'
: Define the size for the colour cache. In PROforma each gstate keeps a few colours which were last used to make sure that the pattern which is used to estimate the colour does not have to be recalculated all the time. This causes a very big speed increase in some operations, especially for drawing pictures. You can choose how many colours are retained in the colour cache. The value is restricted to stay inside the 1..256 range. The

default value is 8.

'R'

load a font file as resident font. A resident font will always remain in memory (unless PROforma is removed). The first resident font is considered to be the built-in font (which is essential for proper functioning). The characters from the built-in font are (also) displayed when that character is not available in the current font. It is therefore recommended that the built-in font be as complete as possible.
The parameter is the name of the fontfile, which is searched on the searchpath for fonts. If you also want to be able to choose the resident fonts in the fontmap, then you should also include a 'P' command.

'P'

this command adds a font to the fontmap. The fontmap is a matching between font names and their filename. The fontmap is also used to figure out which fonts are available. The command has two parameters, separated by a semicolon (';'), there should be no spaces before and after the semicolon. The first parameter is the name of the font (which has to be an exact match, including case). The second parameter is the name of the font file. PROforma font files normally end in '`_pff`'.

'd'

selects the default printer driver. The driver can be given either as the driverid number (in ASCII, this starts with a minus sign as driverid's are negative) or as the full (case sensitive) printer driver name.

The configuration constants are only passed to the last loaded PROforma driver (or if you just selected the default printer driver, than that driver will get the configuration constants). Most printer drivers will normally understand the following configuration constants :

DEFAULT-DEVICE

The parameter if the default device for the printer driver. Some examples are `ser1hr` or `pard`. Note that PROforma only prints raw data, so translates should be switched off, hence the '`r`' in `ser1hr` and the '`d`' in `pard`.

PRINTABLE-AREA-SIZE

Allows you to set the size of the *printable area* for your printer. This is the area where output can be visible on the page. The parameters are in

typographical points, which has a unit if 1/72 inch or approx. .35 mm.

PRINTABLE-AREA-ORIGIN

Allows you to set the origin of the *printable area* for your printer, or to put it differently, the offset of the printable area from the left and top of the page. The parameters are in typographical points, which has a unit if 1/72 inch or approx. .35 mm.

For example, if you want to configure the epson compatible 9 pins printer driver to have margins of one inch at each side, use 8.5x11 inch paper, and print to the serial port by default, then part of your PROforma_cfg should look like this :

```
D Epson9_pfd
PRINTABLE-AREA-SIZE 468 648
PRINTABLE-AREA-ORIGIN 72 72
DEFAULT-DEVICE ser1hr
```

boot file

When ProWesS is started, three files are used to give instructions about which files have to be loaded etc. These three files are the boot file, the ProWesS startup file ($PWSDIR_startup, where $PWSDIR is your ProWesS directory), and a file with your personal configuration ($PWSDIR_personal_ldr). Of these, you can both change the `boot` and `personal_ldr` files to customize your system.

These files all have a specific usage :

- The *boot* file is used to load all the extensions which are necessary start the loading process of ProWesS and all the SuperBASIC extensions (as these can't be loaded in the other files). Apart from this some other system initialisations are also done in the boot file like setting the display resolution and colour depth, assigning hard drive numbers, setting hotkeys,...
- The *startup* file is used by ProWesS to load all the extensions that ProWesS needs to work properly. It also loads some programs which are assumed to be in memory by some of the ProWesS utilities. This file should not be modified.

- The *personal_ldr* file is started by the startup file and contains your local customized info. This is the place where the buttons which have to be displayed are defined, and where printer drivers are added, fonts are made available and where extra applications can be installed.
  When you install a program, some lines are usually added to this file (as it is easier to handle than the boot file). You can also edit this file by hand.

The boot file which is generated by ProWesS is rather generic. If you want to have other resident extensions and programs, then this is a good place to add them. The boot file contains many comments with guidelines about where to put which sort of commands. Just use the boot file as a template to customize your system.

buttons in ProWesS

The default configuration of ProWesS displays two buttons and a clock. This can easily be modified. The `mine` subdirectory in ProWesS contains your personal configurations. The buttons etc. which are started are defined in the `personal_ldr` file. A button is defined by executing cbutton. This program will create a button in the button frame. If the button is indicated with a *DO*, then the ProWesS loader is started with the given filename as parameter. If that file is an executable, then that file is executed, otherwise, the commands in that file are executed.

For example, the following lines, when added in your personal_ldr file, create a button which will execute the *make* program (to compile a program) :

```
%% add button for make
cbutton -name "compile all" win1_c68_make
```

Unfortunately, your screen would soon be too small if all applications would require their own button. So there is a utility program (mbutton), which allows you to group several items in one menu.

The *utilities* button for example is defined as follows :

```
cbutton -name "utilities" utilities_ldr
```

The `utilities_ldr` file just calls *mbutton* to display the menu.

mbutton utilities_mbt -name utilities

The items which have to be displayed inside the menu are defined in the `utilities_mbt` file. Each line in the file is an item in the menu. The line can contain the name which has to be displayed, the loader file or executable which has to be called, and the searchpath for the file. The file used above is given here.

```
install -path flp1_ -name "start program in flp1_"
procon -path $PWSDIR_prg -name "configure (procon) (level2)"
PWconfig -path $PWSDIR_prg -name "configure PROforma/ProWesS"
calc -name "calculator"
multiview -path $PWSDIR_prg -name "file viewer"
global -path $PWSDIR_prg -name "Global Variables"
install -path $PWSDIR_prg -name "install software"
```

Frequently asked questions

- **Why is ProWesS not included with the programs which need it ?**

  ProWesS is a very powerful system. It is distributed as a separate package because the entire package is quite big. If you would want to include the ProWesS package with each copy of LINEdesign or DATAdesign, then you would always need an extra disk.

  However, the question is all wrong. It suggests that you consider ProWesS to be the same as the Pointer Environment. However, ProWesS does a lot more, it provides more libraries (not just a window manager), and also includes lots of extra support software for installing programs, displaying and printing documentation etc. The end result is that ProWesS allows the software producers to deliver the software to you cheaper and faster than before. Also, once you get the program, using it is a lot easier. There is no need to modify the boot file, there is no need to reset the computer to start the program. Even if you are not a very experienced QL user, you can still easily multitask all the programs etc.

- **Why do ProWesS and the ProWesS applications need installation ?**

  One of the main goals for ProWesS is to make your computer easier to use. Since the release of the QL, it was built to multitask several programs. However, many programs need some extensions to be loaded, and therefore, many users had no other option but to reset the computer and boot up with a specific program. It is only once you can modify and write boot files yourself that this can be solved.

  In ProWesS this is no longer true. When you install ProWesS, it can build a boot file for you. After that, modifying the boot file is never necessary (though you can customize it a lot, and guidelines for this are given in the file). Most programs can be started by indicating *start program in flp1_* in the *utilities* button. Even better, you can install the

applications so that they appear in a button on screen. Using your computer has never been easier.

- **How can I print out a manual in ProWesS ?**

  To print out a manual, you have to start the ProWesS reader. This is normally possible by indicating *read documentation* in the *applications* button. The ProWesS reader includes a *print* command which allows you to print the document which is loaded and all the subdocuments (when that option is indicated).

  To be able to print an entire set of documentation, you should start from the *table of contents* file. This is usually called either *toc.html* or prg-name.*html*.

  To be able to print a document, a printer driver has to be available. You can add (and configure) printer drivers in your system by using the *configure PROforma* option in the *utilities* button.

- **How can I get an update for ProWesS ?**

  You can get a ProWesS update directly from PROGS or from your local dealer. When not ordering anything else you probably have to pay postage and possibly a small handling fee. You have to send your master disk(s) to get the update.

  For registered users, there is also a much quicker way. Updates to ProWesS are regularly posted on Jochen Merz's BBS. If you are a registered ProWesS user, you can get access to the *ProWesS updates* file area (60). There you can get zip files which contain all the changes since the full release version. You have to unzip these files on top of your master disk(s) and then install ProWesS again. The installation options which should be indicated are *update installation* and possibly *install documentation* if you want the docs to be updated.

  We strongly advice to get ProWesS updates regularly. Some ProWesS applications need fairly recent versions of ProWesS to operate. Also

updates can remove bugs and increase the speed, which is interesting for all programs.

- **Why does it take so long for the ProWesS buttons to appear ?**

  ProWesS is by default configured to precalculate the fonts which are used in the menus. This takes only a limited amount of memory and makes sure that windows are always drawn as quickly as possible. This uses the `SYSTEM-FONT-CALCULATED` config definition in ProWesS. You can remove these tags to make sure this delay does not happen, but this is not adviseable. When the screen fonts are not precalculated, drawing windows will take a while before the characters are all in the cache. The drawing can also slow down again after you have printed something (as the letters cached characters will be replaced).

ProWesS reader

PROGS, Professional & Graphical Software
Dr. Frans Hemerijckxlaan 13 /1
2650 Edegem
BELGIUM
tel : +32 (0)3/ 457 84 88 fax : +32 (0)3/ 458 62 07 e-mail : joachim@club.innet.be
www : http://www.club.innet.be/~year2827

Introduction

The ProWesS reader is a program which can display hypertext documentation. These are text files with some hints about proper display, and possible links between documents. The file format which is used for this is HTML, the *HyperText Markup Language*.

This program is mainly intended for online reading of documents, especially manuals and help files. However, it can also produce hardcopy of these documents.

The ProWesS reader is normally used for reading hypertext documents. These documents often consist of several files, which have links between them. To be able to fully use hypertext documents, you usually need to start at the *main menu* or *table of contents*. These can usually be found in a file ending in `_toc.html` or a file which has the program name as file name (e.g. this file : `reader.html`).

Menu Bar

The menu bar is the list of items just below the title bar (which contains the help, quit and sleep items), and just above the items which display the name of the document being viewed.

Although it is possible to change all the item names, I will use the default names for reference. All the items can also be indicated by pressing a key. The key which should be pressed is the first letter of the (default) item name, except for *bookmarks*, which can be indicated by pressing <o>.

Back

The *Back* items allows you to go back on your tracks. It will redisplay the file which was being viewed just before this one.

Load

Select the file which has to be displayed. A file select window is displayed to allow you to make your choice interactively. Please note that the ProWesS reader can only display HTML hypertext documents. Other files could give unexpected results.

Reload

This item will load the file which is currently displayed again. This can be useful when you resized the window (as the line width is not automatically changed).

More importantly, this allows you to reload the page easily when you are designing or modifying the contents.

Print

To allow you to read a big part of documentation more easily, you can also produce proper hardcopy by printing them on your printer. To make things readable, you should set the fontsize to a value which is suitable for printing. Values which are often used are eleven (11) or twelve (12), which match the sizes often used in books.

Of course, the *device* and *printer driver* which should be used for printing can be set. Please note that the ProWesS reader will try to make full use of the printable area of your printer. To make sure that nothing falls of the edge, your printer driver should be properly configured.

The ProWesS reader will not just print the document which is being displayed, but will try to include all subdocuments in the printout (without duplicates of course). This is done by giving hints when writing the HTML documents. All references which include the `PRINT`, `REV="toC"` or `REL="SUBDOCUMENT"` attribute in the reference will be included.

If you want to punch holes in the documents you are going to print, then the *Margin for perforation* option should be indicated. This will make sure that a one inch margin is available at the left side of the page.

Save the trees ! Remember that you can directly reuse a lot of paper by also printing on the backside of pages which were used at the other side only. This can save a lot of trees and energy on recycling. Please do not throw used paper in the bin, but have it recycled, our world has to survive for many generations to come !

Styles

This menu allows you to change the (high level) style which is used for the document, like the fontsize which has to be used on screen, and the selection of fonts. Many more can be changed by configuring the program (see config).

Please note that the *typewriter font* should be a mono spaced (or fixed width) font. A lot of documents representation depends on all the characters having the same width in this font. A typical example of a mono spaced font is the *Courier* font family.

Bookmarks

It is possible to define *Bookmarks*. This is a list of files which you often need, so that you can jump to them directly. The list of bookmarks is loaded from a file when the ProWesS reader is started. The name of the bookmarks file can be configured (see config).

The bookmarks menu contains the bookmarks which can be used and three items

Add current
: This adds the file which is currently displayed to the bookmarks list (at the bottom). Please note that this can generate duplicates.

Load
: This command will display the file select window, so that you can indicate a new bookmarks file. An error will be reported if the file which was indicated is not a bookmarks file.

Save
: When you indicate this window, a window is displayed in which you can edit the filename to save the bookmarks file. You can still cancel by indicating *quit* or pressing <esc>.
In this window you cannot move the pointer using the cursor keys as they control the cursor in the edline object. The filename and save action are confirmed by pressing <enter>.

The bookmarks file is a human readable file. This allows you to load it in an editor and manually sort or delete items (each line is an item from the bookmarks list).

History

The *History* window gives the list of the files which were last displayed. This

allows you to go back to one of the previous files in somewhat larger steps than by using *Back*.

Configuration

As could be expected, The ProWesS reader is quite configurable. The things which are configurable are be divided into three parts.

General options

- default window size
- page colour scheme (white on black or black on white)
- the file which contains the bookmarks
- default printer driver
- directory with the ProWesS reader documentation

Language options

- All the item names and the labels which are used in the ProWesS reader are configurable.

Stylepage

- All the items from the *Style* menu can be configured here, and *much more*. The entire look of documents can be configured. All the sizes etc. are relative to the body font size. This makes the look scale along nicely when the fontsize is changed (e.g. when printing).

  The body font size can be configured to zero. In that case the ProWesS default fontsize will be used.

Command line options

`reader [-help] [-file filename] [-pos position] [-dir directory]`

help
: This option indicates that the ProWesS reader is used to display the help files. In this case the *Load* and *Bookmarks* items are not available.

filename
: This allows you to pass the name of the file which has to be displayed

when the program starts.

position

The name of the position in the file which should be displayed.

directory

The directory where the file should be searched.

HTML - Hypertext Markup Language

Introduction

HTML is an application of SGML, the *Standard General Markup Language*. It has the shape of text which is enriched with extra *markup*. Two kinds of markup are possible :

tags
: A tag is a name enclosed in angled brackets (< and >).Tags are normally encountered in pairs, a start tag (just the name in backets), and an end tag (the name is preceded by a slash (/)).
For example `<I>italics</I>`.
Tags can also have extra attributes, which can be given after the name, but before the closing bracket.
Tags which are not recognised are skipped !

entities
: To allow access to characters which are not always available in the standard character set on a computer, and to allow access to reserved characters (like < and >, see above), entities are also allowed. An entity denotes character, and has to be given by name. Entities are preceded by an ampersand, and ended by a semicolon (e.g. `&copy;` for ©).
For a list of the possible entities, click here.

In HMTL documents, whitespace is mostly skipped. Line breaks which exist in the source document are translated to whitespace, and all whitespace is just rendered as one word spacing. This allows you to make the source document look good, without affecting the final rendering. However, this changes in the *PRE* element, which will display preformatted text, and maintains the organisation the source (see section Preformatted Text: PRE).

Document Structure

An HTML document is a tree of elements, including a head and body,

headings, paragraphs, lists, etc.

Document Element: HTML

The HTML document element consists of a head and a body, much like a memo or a mail message. The head contains the title and optional elements. The body is a text flow consisting of paragraphs, lists, and other elements.

Head: HEAD

The head of an HTML document is an unordered collection of information about the document. For example:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HEAD>
<TITLE>Introduction to HTML</TITLE>
</HEAD>
...
```

Title: TITLE

Every HTML document must contain a *TITLE* element.

The title should identify the contents of the document in a global context. A short title, such as "Introduction" may be meaningless out of context. A title such as "Introduction to HTML Elements" is more appropriate. (Although the length of titles is not limited, long titles may truncated in some applications. To minimize this possibility, titles should be limited to less than 64 characters. The ProWesS reader has a maximum title length of 80 characters).

The ProWesS reader uses the title of a document both in the history list and as a label for the window displaying the document. This differs from headings (section Headings: H1 ... H6), which are typically displayed within the body text flow.

Body: BODY

The *BODY* element contains the text flow of the document, including headings, paragraphs, lists, etc.

For example:

```
<BODY>
<h1>Important Stuff</h1>
<p>Explanation about important stuff...
</BODY>
```

Headings: H1 ... H6

The six heading elements, *H1* through *H6*, denote section headings. Although the order and occurrence of headings is not constrained by HTML, it is advised not to skip levels (for example, from H1 to H3), as converting such documents to other representations is often problematic.

Example of use:

```
<H1>This is a heading</H1>
Here is some text
<H2>Second level heading</H2>
Here is some more text.
```

Typical renderings are:

H1
: Bold, very-large font, centered. One or two blank lines above and below.

H2
: Bold, large font, flush-left. One or two blank lines above and below.

H3
: Italic, large font, slightly indented from the left margin. One or two blank lines above and below.

H4
: Bold, normal font, indented more than H3. One blank line above and below.

H5
: Italic, normal font, indented as H4. One blank line above.

H6

Bold, indented same as normal text, more than H5. One blank line above.

Block Structuring Elements

Block structuring elements include paragraphs, lists, and block quotes. They must not contain heading elements, but they may contain phrase markup, and in some cases, they may be nested.

Paragraph: P

The *P* element indicates a paragraph. The exact indentation, leading space, etc. of a paragraph is not specified and may be a function of other tags, style sheets, etc.

Typically, paragraphs are surrounded by a vertical space of one line or half a line. The first line in a paragraph is indented in some cases.

Example of use:

```
<H1>This Heading Precedes the Paragraph</H1>
<P>This is the text of the first paragraph.
<P>This is the text of the second paragraph. Although you do not
need to start paragraphs on new lines, maintaining this
convention facilitates document maintenance.</P>
<P>This is the text of a third paragraph.</P>
```

Preformatted Text: PRE

The *PRE* element represents a character cell block of text and is suitable for text that has been formatted for a monospaced font.

Within preformatted text:

- Line breaks within the text are rendered as a move to the beginning of the next line.
- Anchor elements and phrase markup may be used.
- Elements that define paragraph formatting (headings, address, etc.) must

not be used.
- The horizontal tab character (code position 9 in the HTML document character set) must be interpreted as the smallest positive nonzero number of spaces which will leave the number of characters so far on the line as a multiple of 8. Documents should not contain tab characters, as they are not supported consistently.

Example of use:

```
<PRE>
Line 1.
       Line 2 is to the right of line 1.     <a href="abc">abc</a
       Line 3 aligns with line 2.            <a href="def">def</a
</PRE>
```

Address: ADDRESS

The *ADDRESS* element contains such information as address, signature and authorship, often at the beginning or end of the body of a document.

Typically, the *ADDRESS* element is rendered in an italic typeface and may be indented.

Example of use:

```
<ADDRESS>
Newsletter editor<BR>
J.R. Brown<BR>
JimquickPost News, Jimquick, CT 01234<BR>
Tel (123) 456 7890
</ADDRESS>
```

Block Quote: BLOCKQUOTE or BQ

The *BLOCKQUOTE* element contains text quoted from another source.

A typical rendering might be a slight extra left and right indent, and/or italic font. The *BLOCKQUOTE* typically provides space above and below the quote.

Single-font rendition may reflect the quotation style of Internet mail by putting a vertical line of graphic characters, such as the greater than symbol (>), in the left margin.

The ProWesS reader allows you (conforming with HTML3) to shorten the *BLOCKQUOTE* tag to *BG*. Also, the rendition is exactly the same as the *ADDRESS* element.

Example of use:

```
I think the play ends
<BLOCKQUOTE>
<P>Soft you now, the fair Ophelia. Nymph, in thy orisons, be all
my sins remembered.
</BLOCKQUOTE>
but I am not sure.
```

List Elements

HTML includes a number of list elements. They may be used in combination; for example, a *OL* may be nested in an *LI* element of a *UL*.

In compliance with HTML3, lists can be provided with a title which is rendered just before the actual list, typically in a somewhat larger font. A list should have at most one title, which should should be given before the list items.

The list header uses the *LH* element. For example :

```
<UL>
<LH>List header</LH>
<LI>List item
<LI>Another list item
</UL>
```

Unordered List: UL, LI

The *UL* represents a list of items -- typically rendered as a bulleted list.

The content of a *UL* element is a sequence of *LI* elements. For example:

```
<UL>
<LI>First list item
<LI>Second list item
 <p>second paragraph of second item
<LI>Third list item
</UL>
```

Ordered List: OL

The *OL* element represents an ordered list of items, sorted by sequence or order of importance. It is typically rendered as a numbered list.

The content of a *OL* element is a sequence of *LI* elements. For example:

```
<OL>
<LI>Click the Web button to open URI window.
<LI>Enter the URI number in the text field of the Open URI
window. The Web document you specified is displayed.
  <ol>
   <li>substep 1
   <li>substep 2
  </ol>
<LI>Click highlighted text to move from one link to another.
</OL>
```

Directory List: DIR

The *DIR* element is similar to the *UL* element. It represents a list of short items, typically up to 20 characters each. Items in a directory list may be arranged in columns, typically 24 characters wide.

The content of a *DIR* element is a sequence of *LI* elements. Nested block elements are not allowed in the content of *DIR* elements. For example:

```
<DIR>
<LI>A-H<LI>I-M
<LI>M-R<LI>S-Z
</DIR>
```

Menu List: MENU

The *MENU* element is a list of items with typically one line per item. The menu list style is typically more compact than the style of an unordered list.

The content of a *MENU* element is a sequence of *LI* elements. Nested block elements are not allowed in the content of *MENU* elements. For example:

```
<MENU>
<LI>First item in the list.
<LI>Second item in the list.
<LI>Third item in the list.
</MENU>
```

Definition List: DL, DT, DD

A definition list is a list of terms and corresponding definitions. Definition lists are typically formatted with the term flush-left and the definition, formatted paragraph style, indented after the term.

The content of a *DL* element is a sequence of *DT* elements and/or *DD* elements, usually in pairs. Multiple *DT* may be paired with a single *DD* element. Documents should not contain multiple consecutive *DD* elements.

Example of use:

```
<DL>
<DT>Term<DD>This is the definition of the first term.
<DT>Term<DD>This is the definition of the second term.
</DL>
```

If the DT term does not fit in the DT column (typically one third of the display area), it may be extended across the page with the DD section moved to the next line, or it may be wrapped onto successive lines of the left hand column.

Phrase Markup

Phrases may be marked up according to idiomatic usage, typographic appearance, or for use as hyperlink anchors.

User agents must render highlighted phrases distinctly from plain text. Additionally, *EM* content must be rendered as distinct from *STRONG* content, and *B* content must rendered as distinct from *I* content.

Phrase elements may be nested within the content of other phrase elements; however, HTML user agents may render nested phrase elements indistinctly from non-nested elements:

```
plain <B>bold <I>italic</I></B> may be rendered
the same as plain <B>bold </B><I>italic</I>
```

Idiomatic Elements

Phrases may be marked up to indicate certain idioms.

Citation: CITE

The *CITE* element is used to indicate the title of a book or other citation. It is typically rendered as italics. For example:

```
He just couldn't get enough of <cite>The Grapes of Wrath</cite>.
```

Code: CODE

The *CODE* element indicates an example of code, typically rendered in a mono-spaced font. The *CODE* element is intended for short words or phrases of code; the *PRE* block structuring element (section Preformatted Text: PRE) is more appropriate for multiple-line listings. For example:

```
The expression <code>x += 1</code>
is short for <code>x = x + 1</code>.
```

Emphasis: EM

The *EM* element indicates an emphasized phrase, typically rendered as italics. For example:

```
A singular subject <em>always</em> takes a singular verb.
```

Keyboard: KBD

The *KBD* element indicates text typed by a user, typically rendered in a mono-spaced font. This is commonly used in instruction manuals. For example:

```
Enter <kbd>FIND IT</kbd> to search the database.
```

Sample: SAMP

The *SAMP* element indicates a sequence of literal characters, typically rendered in a mono-spaced font. For example:

```
The only word containing the letters <samp>mt</samp> is dreamt.
```

Strong Emphasis: STRONG

The *STRONG* element indicates strong emphasis, typically rendered in bold. For example:

```
<strong>STOP</strong>, or I'll say "<strong>STOP</strong>" again!
```

Variable: VAR

The *VAR* element indicates a placeholder variable, typically rendered as italic. For example:

```
Type <SAMP>html-check <VAR>file</VAR> | more</SAMP>
to check <VAR>file</VAR> for markup errors.
```

Typographic Elements

Typographic elements are used to specify the format of marked text.

Typical renderings for idiomatic elements may vary between user agents. If a specific rendering is necessary -- for example, when referring to a specific text attribute as in "The italic parts are mandatory" -- a typographic element can be used to ensure that the intended typography is used where possible.

Bold: B

The *B* element indicates bold text. Where bold typography is unavailable, an alternative representation may be used.

Italic: I

The *I* element indicates italic text. Where italic typography is unavailable, an alternative representation may be used.

Teletype: TT

The *TT* element indicates teletype (monospaced)text. Where a teletype font is unavailable, an alternative representation may be used.

Anchor: A

The *A* element indicates a *hyperlink anchor*. At least one of the *NAME* and *HREF* attributes should be present. Attributes of the *A* element :

HREF
: gives the URI of the head anchor of a hyperlink. The ProWesS reader can only access local documents, so this makes the URI quite limited. You can reference files by giving the filename as value. The file will be searched in the same directory as the current file. If you want, a position in the file can also be given. This position is given by *name*, just after the filename, separated by a hash (#). For local links, the filename should be omitted. For example :

```
<A HREF="myfile.html">external link</A>
<A HREF="#somewhere">local link</A>
<A HREF="myfile.html#somewhere">position in file</A>
```

: All dots and (back)slashes in the filenames are translated to underscores by the ProWesS reader. This allows access of external HTML documents.

NAME
: gives the name of the anchor, and makes it available as a head of a

hyperlink.

REL

The *REL* attribute gives the relationship(s) described by the hyperlink. The value is a whitespace separated list of relationship names. The semantics of link relationships are not specified in this document.
The ProWesS reader will include referenced objects in printout if the *REL=SUBDOCUMENT* attribute/value pair if found (value is compared case independant).

REV

same as the REL attribute, but the semantics of the relationship are in the reverse direction. A link from A to B with REL="X" expresses the same relationship as a link from B to A with REV="X". An anchor may have both REL and REV attributes.
The ProWesS reader will include referenced objects in printout if the *REV="toC"* attribute/value pair if found (value is compared case independant).

PRINT

the ProWesS reader also supports an extra *PRINT* attribute value, which indicates that the referenced document should also be printed when the user requests hardcopy of the document.

Line Break: BR

The *BR* element specifies a line break between words. For example:

```
<P> Pease porridge hot<BR>
Pease porridge cold<BR>
Pease porridge in the pot<BR>
Nine days old.
```

Horizontal Rule: HR

The *HR* element is a divider between sections of text; typically a full width horizontal rule or equivalent graphic. For example:

```
<HR>

<ADDRESS>February 8, 1995, CERN</ADDRESS>
```

```
</BODY>
```

Image: IMG

The *IMG* element refers to an image or icon via a hyperlink.

HTML user agents may process the value of the *ALT* attribute as an alternative to processing the image resource indicated by the *SRC* attribute.

Attributes of the *IMG* element :

ALT
: text to use in place of the referenced image resource, for example due to processing constraints or user preference.

SRC
: specifies the URI of the image resource.

UNITS
: Give the unit which is used in the value of the *WIDTH* and *HEIGHT* attributes. The possible values are *PIXELS* or *EN*. The default is pixels. An en is half the point size which is in use. The ProWesS reader also accepts *CW* as unit. One cw equals the current width of the column.

WIDTH
: Specify the width for the image.

HEIGHT
: Specify the height for the image.

In the ProWesS reader, all images are always displayed flush left on a separate line. To be able to display a picture, there has to be a PROforma picture type which can recognize and display that picture. When the size to display the picture is not given, then the picture will get a width of half the width of the area in which the document is displayed (as if you included `WIDTH=.5 UNITS=CW`. If you only specify either the width or the height of the picture, then the aspect ratio of the picture will be retained. When the `UNITS=EN` attribute is given, then one en is half the *current* fontsize. Because the ProWesS reader also has to be able to print HTML files properly, the picture size when given in pixels is approximated by using points (as if you are running at a resolution of 720 by 540).

Examples of use:

```
car1_com
<img src="win2_tmp_car1_com">
car2_com
<img src="win2_tmp_car2_com" height=10 units=en>
```

This document is mostly based on a part of the HTML 2.0 specification
PROGS, Professional & Graphical Software
last edited September 27, 1996

ProWesS loader

PROGS, Professional & Graphical Software
Dr. Frans Hemerijckxlaan 13 /1
2650 Edegem
BELGIUM
tel : +32 (0)3/ 457 84 88 fax : +32 (0)3/ 458 62 07 e-mail : joachim@club.innet.be
www : http://www.club.innet.be/~year2827

Introduction

The ProWesS loader is a utility program for ProWesS. It is used to start applications with full configuration and based on a *startup file* which can make sure that all the necessary extensions are installed.

The loader will execute the commands which are given in the startup files. The commands themselves can either be executable things, or will load them on the searchpath. The searchpath is also given as parameter to each of the commands, preceded by '`-path`'.

If the startup file which has been passed happens to be an executable, then that program will be executed (without passing any parameters to it).

Usage

`loader` *`startup-file`* `[-path` *`path`*`]`

startup-file
: The program needs one parameter when it is started, the name of the *startup file*. Startup files are normally called "`startup`", or have a name ending in "`_ldr`".

path
: This option allows you to give the path where the loader file has to be searched. If you do not tell the program where to look, the file will be searched first on the data default and then the program default device.

Input File Format

The file format is line oriented. Each line contains a command. However, if the command and the parameters do not fit on one line, then you can make sure that the next line is appended by ending the line with a backslash. Empty lines and lines starting with a percentage sign or a semicolon ('`%`' or '`;`') are discarded as comments. Lines which start with two precentage signs ('`%%`') will be used by maintenance programs to customize startup files.

Lines starting with an ampersand ('`&`'), followed by a program name (not separated), and then the parameters will execute thing or file with that name and wait for the result (even the program name can contain spaces when enclosed in double quotes).

Lines starting with an asteriks ('`*`'), followed by the name of a startup file (not separated) will run a different startup file (similar to include, except that the search path is redetermined (and restored afterwards).

Lines starting with plus sign ('`+`'), followed by a searchpath will replace the searchpath by the directory of the startup file, followed by the searchpath which is given as parameter.

Lines starting with a program name (not separated), and then the parameters will execute thing or file with that name and wait for the result (even the program name can contain spaces when enclosed in double quotes).

Some action programs for ProWesS loader
request

Request confirmation from the user (e.g. "change disks").

rext
: Load a resident extension, or a program as executable thing (if the program supports that).

setenv
: Set a "Global Variable", similar to environmnent variables on other systems.

wait
: Wait for a thing to be available, or a fixed time (whichever happens first).

cbutton
: Create a button which will load a program when indicated.

mbutton
: A program which allows you to indicate from a list, which program has to be loaded.

Examples

Set an environment variable. The first word is the program which has to be called. The rest are parameters.

```
setenv PWSDIR win1_pws
```

Parameters can contain spaces by enclosing them in quotes. The ProWesS loader can wait until the program has finished.

```
&request "Please insert another disk"
```

Program names can also contain spaces.

```
"ProWesS reader" -file myfile
```

Commands and their parameters can be spread over several lines.

```
request "just to display that multiple" \
        "lines can be used" \
        "indicate OK"
```

PROGS, Professional & Graphical Software
last edited April 16, 1996

request

Introduction

`request` is a program to request the user to *do* or *confirm* something. The most typical example is the request a insert a certain disk.

The actual message can be given as parameter. Apart from that, the window will contain an *OK* which has to be indicated by the user for the program to terminate.

Usage

`request {`*`line`*`}`

line
: The given line will be displayed in the request window. Several lines can be given. If the line contains spaces, then it has to be enclosed in double quotes ('"'), and the double quotes themselves have to be doubled.

PROGS, Professional & Graphical Software
last edited January 26, 1996

rext

Introduction

`rext` is a program to load a resident extension.

Please note that this program is not suitable for loading programs which link in SuperBASIC extensions. This can only be done by the system !

Usage

`rext` *`filename`* `[-path` *`path`*`]`

filename
: The name of the file which has to be loaded. If no *path* is given, then file will be searched on the program default device.

path
: A path can be given. The file will be searched on the path given. A path can include several device_directory combinations, each separated by a semicolon (';'). A path can also include directories which start with the name of a "Global Variable".

---

setenv

Introduction

`setenv` is a program to set a *Global Variable*. A Global Variable is somewhat similar to *Environment Variables* on some other systems (hence the name). However, this variant really is global, and can be accessed and changed by everybody.

Usage

`setenv` *`name`* [*`value`*] [`-path` *`path`*]

name
: The name of the Global Variable which gets a value.

value
: The value for the given *name*. If the value contains spaces, then it should be included in double quotes (as usual).

path
: If no *value* is given explicitly, then the given path will be used as value. This is done to allow the directory which contains a program to be set automatically (without ever needing configuration) by including `setenv NAME` at the very start of the loader file.

PROGS, Professional & Graphical Software
last edited January 30, 1996

wait

Introduction

`wait` is a program which waits for either a thing to exist, or for a specified amount of time (whichever occurs first).

Usage

`wait [`*`thingname`*`] [-wait `*`period`*`]`

thingname
: The name of the thing which should exist.

period
: Specify the maximum waiting period. The unit is approximately 1/50 of a second. If no maximum waiting period is given, the program will wait forever, until the given thing exists.

PROGS, Professional & Graphical Software
last edited January 30, 1996

cbutton

Introduction

`cbutton` is a program to create a button in the *button frame*. This button allows you to start a program (cbutton stands for "Call BUTTON").

If the *button frame* does not exist, the program will just display a moveable window. The further operations are the same.

When the item in the *cbutton* program is indicated by a *DO*, then a given program will be started. This is done by starting the ProWesS loader which will load that program.

If needed, you can make sure that the program asks you to insert the proper disk before the program is actually started.

To function properly, `cbutton` assumes that the `request` and `ProWesS loader` executable things are available. These are normally loaded when ProWesS is started.

Usage

`cbutton` *`filename`* `[-path` *`path`*`] [-insert] [-name` *`button-name`*`]`

filename
: The name of the file which has to be used by the ProWesS loader when the button is activated. If no *path* is given, then file will be searched on the data default device (and then on the program default device).

path
: A path can be given. The file will be searched on the path given. A path can include several device_directory combinations, each separated by a semicolon (';'). A path can also include directories which start with the name of a "Global Variable".

insert

When this option is given, the program will ask the user to insert a disk before it tries to load the program.

button-name

You can explicitly tell the button which name has to be displayed inside it. If you do not specify the button name, then the filename will be displayed.

PROGS, Professional & Graphical Software
last edited March 30, 1996

mbutton

Introduction

`mbutton` is a program which displays a list of "applications" which can be loaded. If you indicate one of the items, then that item will be loaded, and the window will be removed.

The list of programs which can be started, comes from a special file which gives details about the name of the choices, the loader file which has to be used to start the application, and possibly also a path where the loader file can be found. If needed, you can also let the program ask for the proper disk to be inserted.

To function properly, `mbutton` assumes that the `request` and `ProWesS loader` executable things are available. These are normally loaded when ProWesS is started.

Usage

`mbutton` *`filename`* `[-path` *`path`*`] [-name` *`name`*`] [-wait]`

filename
: The name of the file which has to be used by the ProWesS loader when the button is activated. If no *path* is given, then file will be searched on the data default device (and then on the program default device).

path
: A path can be given. The file will be searched on the path given. A path can include several device_directory combinations, each separated by a semicolon (';'). A path can also include directories which start with the name of a "Global Variable".

name
: Give a name to the mbutton program You can explicitly tell the button which name has to be displayed inside it. If you do not specify the button name, then the filename will be displayed.

wait
: When this option is given, then the mbutton program will not terminate after starting the ProWesS loader. Therefore, the user can indicate more than one of the options.

Input file format

Each line in the input file, corresponds with a program which can be selected in the window. The length of each line is limited to be less than 160 characters.

Each line contains several tokens. Normally each word is a token, but words can also be grouped by putting them in double quotes. A token which starts with a dash (`-`) indicates a flag. If that flag has a parameter, then the next token is used as that parameter.

Each line should contain a token which indicates the name of the file which is used by the loader program to start the program. If several tokens exist which are not flags and not a parameter to a flag, then the last one will be used as filename.

Several flags are supported by `mbutton`

-insert
: This indicates that the user is asked to insert the proper disk before the ProWesS loader is started to load the application program. this flag has no parameter.

-terminate After the loader file has been executed, mbutton will terminate, even if the `-wait` flag was passed on the command line.

-path
: Specify the path where the ProWesS loader should try to find the loader file. If this is not given, mbutton defaults to first searching on the data device and then on the program device.

-name
: Specify the name for this program, which has to be displayed in the menu. If this is not explicitly given, then the filename of the loader file will be used as application name.

ProWesS calculator

PROGS, Professional & Graphical Software
Dr. Frans Hemerijckxlaan 13 /1
2650 Edegem
BELGIUM
tel : +32 (0)3/ 457 84 88 fax : +32 (0)3/ 458 62 07 e-mail : joachim@club.innet.be
www : http://www.club.innet.be/~year2827

Calculating

The ProWesS calculator allows you to do some simple calculations. You can type the numbers or indicate them, and add, multiply, subtract or divide them. The calculations occur in while typing (so precedence rules are not considered).

The current subtotal is displayed in the display window. The fontsize and colour of the display window can be configured.

The result can be obtained by pressing <=>, or (if the pointer is not inside an item) by pressing <enter>.

The display can be cleared by indicating the "Clear" item, or by pressing <esc>. However, if you press <esc> when the display is already cleared, the program will be terminated.

It is possible (especially in binary representation) that not all digits of a number can be displayed in the window, in that case you should resize the

window to make everything visible.

Base Conversions

The ProWesS calculator is also capable of doing base conversions. The base which is in use is always indicated.

When the base is changed, the value in the display window is converted, and all items for the digits which can not be used are made unavailable.

Changing base is possible by indicating the item or pressing <x> for hexadecimal representation, <t> (cfr. ten) for decimal representation or <i> for binary representation.

Configuration

The ProWesS calculator is very configurable. It contains a configuration block which allows configuring with any level 2 config program (sorry, there is no such program which is part of ProWesS just yet, it is possible using the *MenuConfig* program which is distributed by *Jochen Merz*).

The following aspect are configurable :

- The names of all the items in the window.
- Font, fontsize and colour of the display window.
- Directory where the manual (this file) can be found.
- Whether the calculator should start full sized or sleeping.
- Whether there should be a difference between <enter> and a *DO*. If there is a difference, then pressing <enter> will always display the result, even if the pointer is inside an item.

PROCON, the ProWesS Configuration program

Introduction

Procon is a configuration program (like Config or Menuconfig) that allows you to configure all files that have standard configuration blocks, whether they are 'level 1' or 'level 2'. These configuration blocks contain the configuration information: the descriptive texts, some other information, and, of course, the configured values - all of these make up the configuration items.

Some programs/files have several configuration blocks (e.g. FiFi), others have only one. Each configuration block has a name, which is supposed to say something about the configuration items it contains.

For programs that have a "level 2" configuration block, the configuration values are also stored in a separate file, called 'INF file' (e.g. menuconf_inf). This allows you to configure a program once, and, if you get a new version of the program, you can **UPDATE** the configuration values of the new version with the values already used in the older version, so that you don't have to go

through the entire configuration proccess again.

Procon can also **LEARN** the current configuration settings of all your programs already configured (if they are level 2 programs), so that it will know them the next time round.

Procon, should be able to handle most, if not all, standard configuration items. It is executed with a simple **EXEC** command.

Once you **EXEC**'d the program in the normal way, it will draw its windows: you can see several items, and two menus, which are empty at first. Procon needs the QLiberator runtimes to be loaded.

Overview of the window

At the top, you have the menu bar, containing the "Save", "Back" and 'Help' items, the name of the program and the "Quit", "Sleep" and "About" items. The use of the latter three is obvious and will not be mentioned further.

Further down comes a line where you can indicate the name of the INF file, and the "Write" item to write the INF file back if it has changed.

Underneath is a row with the four main items, which are are "Update", "Learn", "Config" and "Files". What they do exactly often depends on the context.

Finally, there are two menus. We shall call the menu to the left the *left menu* and the menu to the right, the *right menu*. Yes, I know, this is quite a feat of imagination.

Loading the INF file

You should first of all load the "INF" file, i.e. the file that contains the configuration information. Generally, this will be called "MenuConf_inf" or "**Procon_inf**" in the Prowess_prg_ directory.

You can work without loading this file since it isn't useful for level 1

programs. Unless you do load it, however, you can't use the Update and Learn items: they stay unavailable until a valid INF file was loaded or created.

To load this file, indicate the "Inf file" item (selection key: 'I'). This pulls down the fileselect window where you can select the file.

If you do not have an INF file yet, and wish to create one, just type in the file name of the future INF file in the fileselect object. If the file doesn't exist, you will be asked whether you want to create this file. OK will create it, ESC abandons this.

Do not give the name of an existing file that is not an INF file - there is no way to check whether a file is really an INF file or not. If you give the name of an existing file, *it is presumed that this is the name of an INF file in the correct format*! The program may crash if this assumption turns out to be incorrect.

The program can also automatically load the INF file that you have configured (see below - Configuration).

Indicating the files to work on

Normally, once you have loaded an INF file, you will attempt to configure one or several files. One of the advantages of Procon is that it allows you to act on several files automatically at the same time.

Thus, once the INF file is loaded, you must indicate on which file(s) you wish to operate. This is done, not surprisingly, with the "Files" item, which lets you select one or several files. The names of the files chosen will be displayed in the *left menu.* As you can see, all of these files are already selected.

You can now do several things with these files: you can update all or some of them, you can learn the settings of all or some of them, or you can configure only one of them.

Learning the settings of all or some of the selected files

You might now want to learn the configuration values of some or all of the files. This is done simply by indicating some or all of the files in the *left menu*, and then hitting the *learn* item. Procon then learns all the configuration values as they are currently set in these files. The INF file is then **automatically written out**, so that these values are preserved for future use.

Updating all or some of the selected files

As mentioned above, updating means taking the configuration values as they are already configured in the INF file, and setting these values in the file. This is useful for new versions of older programs, where you want to be sure that the new version is configured exactly like the older version.

Indicating *Update* will update all of the selected files. This is why all of the files are already selected. If you do not wish to update one of these files, please deselect it before you Update.

If you update some or all files, the updated files are automatically saved (i.e. overwritten) with the new, updated version. This is done according to the rules for saving files.

Sometimes, when updating a file, Procon will notice that the program being updated contains some new configuration items, which are not yet in the INF file (and which it thus can't update, of course). It will then tell you that there are some new configuration items, and ask you whether it should learn them. If yes, it learns these values as they currently are set. This also warns you that you might want to configure this file explicitly, to search for the unknown configuration items and set values which suit you (don't forget to learn them afterwards!).

Updating all files in a directory

There is also a possibility to update all files contained in a directory: If you start Procon and pass it a directory name in the command line (eg. **EX**

`_procon_obj;"my_dir"`) Procon will automatically update all updatable files in that directory. Please note that the INF file must be found in the file that is preconfigured within Procon. You can configure Procon itself so that it automatically quits once it has configured all (configurable) files in the directory (see below, Configuration).

When doing this, Procon will NOT warn you if some files have new configuration items, which are not yet in the INF file, but it doesn't Learn these new values, either.

If you just EXEC Procon and pass it the string "/a", it will automatically update all programs/files found in the preconfigured directory.

Procon does NOT recurse into subdirectories.

Configuring a single file

Once you have selected several files to work on, you can also choose to configure one single file of these. Indicating the Config item will let you configure the first file selected in the *left menu*. Procon will then change the content of the *left menu* and use it to display "FILE INFORMATION", i.e. the name(s) of the configuration block(s) of this file. If you want to go back to the list of files, hit the "Back" item (selection key: 'B'). (When going back, the file that you just configured will be deselected, so that you can hit Config again for the first file selected etc...)

The configuration information of the first configuration block will also be displayed in the *right menu*.

You can now choose any configuration block simply by indicating it. The configuration items of this block will then be displayed in the *right menu*, and indicating any of these items will bring up a further window which allows you to set the configuration value for this item.

The UPDATE and LEARN items become available if the file contains level 2 configuration blocks - if it contains level 1 blocks, you cannot update or learn. The updating and learning here are done according to the following

rules:

Rules for updating a single file

The *Update* item has the following functions, which depend on where exactly you are when you try to update:

- If no configuration block is selected in the *left menu*, then this updates all the configuration blocks of this file. It does not save the file automatically. You can save the file with the "Save" item.
- If one or several configuration blocks are selected in the *left menu* then Procon updates only the blocks selected. It does not save automatically. You can save the file again with the "Save" item.

It may happen that, when updating, there are some new configuration options. You will be told when this happens - in this case you should later configure the program again, to configure (and learn) the new configuration items.

Rules for learning a single file

Learning is when the software takes the configuration settings in the program from which it learns, and incorporates them in the INF file, so that they may later be used for updating. Learning must ALWAYS be done explicitly. Simply configuring a program **is not enough.** The learn item has the following functions, which depend on where exactly you are when you try to update:

- If no configuration block is selected in the *left menu*, then Procon learns from all the configuration blocks of this file. It does not write out the INF file automatically. You can do that with the "Write" item.
- If one or several configuration blocks are selected in the *left menu* then Procon learns only from the blocks selected. It does not write out the INF file automatically. You can do that with the "Write" item.

For compatibility reasons, Procon uses the same type of INF file as

MenuConfig. One small pitfall of this is that an INF file can only hold the configuration information of a determined and finite number of configuration items. It can thus happen that, whilst learning, the program comes up with an error, telling you that you have reached the upper limit of items in the INF file. If that happens, you should write the INF file out, choose a new INF file and relearn all of the items of the program that was being learned whilst this happened.

Rules for saving a file

The save item works as follows: if you are configuring a determined file, it saves only that file. If not and you have selected some filenames in the *left menu*, indicating the Save item saves all the files that are selected.

Some files are a bit special: they contain a special marker within them, which tells a configuration program such as Procon that they may be saved without the configuration blocks. After all, once you configured the program you shouldn't need the configuration blocks (and the description texts etc) any more, you only need the configured values themselves. Saving the files without the configuration details means that the files will be smaller, but they cannot be configured again later (consequently, never use this option on an original, ALWAYS on a copy)!

Procon notices this special marker, and, where applicable, it will ask before saving whether you wish to save the file without the configuration details. If you say 'no', the file will be saved normlly. If you say yes, you will be given the chance to give a new name to the file, to make sure that you don't overwrite a valuable orginal!

This feature is disabled when Procon updates all files in a directory since you probably don't want it to pop up a window and aks you questions during such an automatic update.

Configuring Procon itself

Procon itself is also configurable:

- You can set the default directory where your programs are. This is preconfigured to *$PWSDIR_prg_*, which is ususally the ProWesS program directory.
- You can set the default name of the _inf file, and whether this should be loaded automatically on startup. Procon comes preconfigured so that the _inf file **Procon_inf** is to be found the the *$PWSDIR_mine_* directory and is loaded automatically.
- Procon can stay after configuring a whole directory: Normally, if you pass a Directory to Procon when you invoke it (i.e. `EX Procon;'win1_whatever_'`), Procon will automatically UPDATE all programs in that directory and then quit - unless you configure it to stay.
- Procon uses a fixed length string for the descriptions/options in its *right menu*. You can set the length of this string: 40 is OK for a small window, 80 for a big window. Procon will try to make its window big enough to contain the longest string in the right hand menu. The preset value is 54.

---

last modified on 20.6.97

PFconfig

PROGS, Professional & Graphical Software
Dr. Frans Hemerijckxlaan 13 /1
2650 Edegem
BELGIUM
tel : +32 (0)3/ 457 84 88 fax : +32 (0)3/ 458 62 07 e-mail : joachim@club.innet.be
www : http://www.club.innet.be/~year2827

Introduction

PFconfig is a program which allows you to fully configure PROforma. The main use is for adding printer drivers or fonts to your system, selecting what the default printer driver is, and to allow you to configure the printable area for your printer.

PFconfig automatically reads the normal config file for PROforma, which is used when your system is booted. When you make a change, then it is effective immediately. You then also get the possibility to make sure the changes you made will also be effective the next time your system boots up.

When PFconfig starts, you have a few options of things which can change. When you indicate such an item or press the first letter of the description,

then you can change that aspect of your PROforma configuration. The item to *make the changes permanent* is only available when something has been changed.

Add driver

You can add new printer, picture and bitmap drivers using this option. A fileselect window will be displayed. In this window you can indicate all the files you want to add in your system.

When you want to save the changes made in this option, then the directory will be added in the searchpath for the drivers, and the drivers themselves are added to the list of drivers. PFconfig will make sure that there are no drivers which are loaded twice and that directories don't occur twice in the searchpath.

Please note that a different driver may be loaded than expected if drivers with the same filename occur in several directories which are in the searchpath for drivers.

Available drivers

printer drivers

- QVME_pfd : the standard screen driver. This is loaded by default.
- LaserJet4_pfd : HP LaserJet 4 driver. Use this if your printer accepts it (it might even work on some newer DeskJets).
- Stylus_pfd : Epson Stylus printer driver. Use this on any ESC/P2 printer.
- StylusColour_pfd : Epson Stylus Colour driver. Uses ESC/P2 in colour.
- DeskJet_pfd : simple HP DeskJet driver. For newer models, use the DeskJet 500 driver, or even (if it works) the LaserJet 4 driver.
- DeskJet500_pfd : HP DeskJet 500 printer driver. Uses better compression than the ordinary DeskJet driver.
- LaserJetIII_pfd, LaserJetII_pfd, LaserJet_pfd : HP LaserJet printer drivers. These support a different set of support commands and

compression. Use the driver with the highest number which works.
- Epson9_pfd : ESC/P printer driver for Epson compatible 9 pin printers.
- Epson24_pfd : ESC/P printer driver for Epson compatible 24 pin printers.
- NecP5_pfd : printer driver for the Nec P5 24 pins printer. The P5 has a different command which is used for vertical spacing, which allows the use of high (360 dpi) resolution).
- bj10_pfd : Canon BJ-10 printer driver. This should also work on other Canon BJ-xxx printers.
- monopic_pfd : printer driver which writes the file to a monochrome pointer environment _pic file. The driver produces output at 288 dpi. You have to specify the filename as the printing device (there is no default). This driver can only output one page.
- fax_pfd : printer driver which writes to a file (name passed as device, always overwritten). The file which is produces is a g3 fax file which can be sent using Qfax. The core routines of this driver have been supplied to us by Jonathan Hudson (thanks for that).

picture drivers

- QLscr_pfd : picture driver which allows the display of standard QL screens, pointer environment _pic pictures (2, 4 and 8 colour) and *The PAINTER* compressed pictures.
- LINEdesign_pfd : allows the display of LINEdesign v2 pictures.
- sprite_pfd : picture driver for mode 4 sprites. This driver has to be loaded if you want to display sprites in your program. This driver is written and copyrighted by Wolfgang Lenerz. Sprite filenames have to end in "_sp4" to be recognized automatically.
- gif_pfd : picture driver to display gif files.

bitmap drivers

- mono_pfd : monochrome bitmap driver. This is used by all the printer drivers, and is loaded by default. It also contains bitmap drivers for multi-plane bitmaps (three and four planes).
- mode4_pfd : mode 4 bitmap driver. This support bitmaps which are

composed as the standard mode 4 screen. It is used by the QVME screen driver, and loaded by default in PROforma.

Configure printer driver

This allows you to configure a printer driver so that the margins on the page are as small as possible without using extra paper, and that the coordinates on screen match the coordinates on paper.

To start, you have to indicate the printer driver which has to be configured. Configurations are always for all the resolutions which are available in one printer driver, so we advice you to choose the highest resolution driver which is available, so as to prevent duplication and overwriting.

When you have selected the printer driver which is to be configured, you get a window which display the name of the driver, the default printing device and the page size which is read from the driver. Unfortunately, PFconfig is not capable of determining what the default device is. To aid you, you can choose the unit which is to be used as metric for the page size and the width of the margins.

At the bottom of the window there are also two rows of items. The first row indicates some default paper size which can be used to set the paper size (and clear the margins). This can be useful when trying to determine the proper settings.

At the bottom row you get some extra items. *Get* can be used to re-read the current settings from the printer driver. It acts as some kind of cancel. You can also *set* the imageable area and default driver of the printer driver to the values which are indicated int he window. The *try* item allows you to test the current configuration of the printer driver.

How to configure your printer driver

The easiest way to configure your printer driver consists of the following steps :

- Set the default device, which is the device your printer is attached to. This is normally either `pard` or `ser1hr`.
- Indicate the paper size which you will use (e.g. A4).
- Indicate *Set* and *Try*. A page will be printed which contains some horizontal and vertical lines. The page also indicates the (normal) distance of these lines with the left/top margin.
- The width of the page should be set to the length of the horizontal lines.
- The height of the page should be set to the length of the vertical lines.
- The margins should be set to the difference between the actual position of the lines and the position which is indicated on the page.
- *Set* should again be indicated to make the proper configuration effective.

It is adviseable that the configuration is tested a few times to make sure that possible differences when feeding the paper can be resolved.

Default printer selection

A window is displayed which displays all the printer drivers which are available in PROforma. You can indicate the printer drivers you want to use as your default driver.

Memory options

It is also possible to configure how PROforma should allocate memory and how much memory should be used for caching. This item will display a menu with the following items.

Max/min memory for Gstate

This allows you to set the maximum amount of memory which PROforma is allowed to use as buffer to render a page. You can also specify a negative amount, in which case at least that amount of memory has to remain free when allocating the buffer.

This item can also be indicated by pressing <m>. The amount are given in bytes.

Size of fontcache

To allow PROforma to display characters as fast as possible, PROforma uses a fontcache. This is a place where characters which have been rendered once are stored to make sure that the image doesn't have to be recalculated each time.

You can set the size of the fontcache. The default is 64kB (65536 bytes), which should be good for most applications. When you haven't got much memory in your system, you could reduce this number (even to zero).

This item can also be indicated by pressing <s>. The amount are given in bytes.

Number of fonts in cache

No matter how big the fontcache is, only a limited number of font/size combinations can be stored in the cache. You can determine what the minimum number of these pairs is. For each Gstate which is open, one extra font/size combination is possible in the fontcache.

Each font/size combination takes about 1.5kB, so you should not make this number too big. On the other hand, when a large fontcache is used, it is adviseable to increase this number as well.

This item can also be indicated by pressing <n>.

Colour cache size

To make switching between colours as fast as possible, PROforma also uses a colour cache. Each Gstate which is opened will allocate some memory in which several colour patterns can be stored. The size of the colour cache is specified as the number of colours which are cached.

This item can also be indicated by pressing <c>.

The colour cache can cause a very large speed increase for some operations. The most important example being the drawing of pictures. The value is restricted to be between one and 256. The default value is 8. If you often use pictures which contain more colours, than it is a good idea to set the colour cache to at least that number.

Note that the pure black and pure white are never stored in the colour cache.

Fonts, add from directory

This option allows you to add fonts to PROforma so that they can be used in your programs. When the item is indicated, a fileselect window is displayed. In this window you have to indicate the font files which you want to be able to access. These fonts will then be made available to PROforma when you indicate *do*.

When the fonts are added in PROforma, the directory where the fonts were found are automatically also added in the searchpath for fonts. The fonts themselves will only be added if no other font with the same name exists.

Search directory for fonts

This menu opion allows you to add extra directories to the searchpath for fonts. All you have to do is type the name of each directory you want in the path, pressing <enter> after each directory. Tou leave the window, press <escape>.

PFConfig will automatically make sure that the searchpath does not contain any duplicates.

Make changes permanent

This option will overwrite the existing `PROforma_cfg` file. It will be replaced by a new file with the info from the previous file, and the changes which have been made.

PROGS, Professional & Graphical Software
last edited December 11, 1996

THE ProWesS AND PROforma CONFIGURATOR

This program can be used to edit the ProWesS and PROforma Configuration files.

First give the program the filename of the configuration file, either by hitting the filename item, and then editing it, or by doing the filename item and selecting the filename from the standard file selector. The file must have the line "**% configuration file for ProWesS**" as its very first line if it is a ProWesS file, or "**; PROforma, PROGS Font & Raster Manager, fontmap config file**" if it is a PROforma file, else it will not be recognized as a valid Configuration file, and the program refuses to load it.

The program automatically recognises whether it is a ProWesS or PROforma configuration file, and the item top left reflects that.

Every line in the file -including comments- is displayed. You can edit each line in turn, except for empty lines, which cannot be edited at all. As soon as you have edited a line, this is passed on to the ProWesS or PROforma internal configurator, for the change to take effect immediately. You can also save the file back, overwriting the older file without any sort of confirmation request.

If you want to get rid of an option altogether, the best way to achieve this is to add a semicolon in front. This way, the option stays in the file and can be reactivated later, but is not taken into account by the ProWesS and PROforma internal configurators.

The "*Add"* item lets you add a new line to the file. There cannot be more than 200 lines.

This program was written with the ProWesS SBasic Interface, and then compiled with QLiberator. There are two versions on the disk - with and without the QLiberator runtimes.

defines for PW_TYPE_APPLIC

APPLIC-SCROLLBAR-LEFT

This constant indicates that the vertical scrollbar has to be displayed to the left of the canvas when present.

APPLIC-SCROLLBAR-RIGHT

This constant indicates that the vertical scrollbar has to be displayed to the right of the canvas when present. This is the default.

APPLIC-SCROLLBAR-ABOVE

This definition constant indicates that if a horizontal scrollbar has to be displayed, it is to be positioned above the canvas which can be scrolled.

APPLIC-SCROLLBAR-BELOW

This definition constant indicates that if a horizontal scrollbar has to be displayed, it is to be positioned above the canvas which can be scrolled. This is the default position.

---

defines for PW_TYPE_DIRSELECT

DIRSELECT-DEVICE

Set a device which should be displayed by each directory select window as an easy to select option to change devices (e.g. "win1_", "flp2_",...)

DIRSELECT-DIRECTORY

Set a directory (including device) which should always be displayed by each directory select window as an easy to select directory. It will be displayed in the window which also lists the subdirectories.

---

defines for PW_TYPE_EDLINE and PW_TYPE_DEDLINE

EDLINE-INK-COLOUR
: Define the RGB colour in which the text in edline objects has to be displayed.

EDLINE-PAPER-COLOUR
: Define the RGB paper colour of the edline objects.

EDLINE-FONT
: Set the PROforma font which should be used by the edline objects.

EDLINE-FONTSIZE
: Set the fontsize in PROforma points which should be used for the text in edline objects.

EDLINE-MAXLENGTH
: Set the default maximum length for the text in edline objects. By default his is 256 (including ending '\0').

EDLINE-ITEMWIDTH
: Set the default minimum size of the edline objects in PROforma points. By default this is 120 (1/6 of the width of the screen).

EDLINE-ALWAYS-TYPE
: This allows you to choose which type of edline should be used. By default, there is a difference between a notmal and a direct edline, but this can be changed. This definition needs a number as parameters. Three values are allowed, 0 (zero) for the default case. When the paremeter is 1 (one), then normal edlines are always used (even when creating a direct edline). When the parameter is 2 (two), then you will always use direct edlines (even when creating the normal variant).

---

defines for PW_TYPE_INFOSTRING

INFOSTRING-INK-COLOUR

Define the RGB colour in which the text in infostring objects has to be displayed.

INFOSTRING-PAPER-COLOUR

Define the RGB paper colour of the infostring objects.

INFOSTRING-FONT

Set the PROforma font which should be used by the infostring objects.

INFOSTRING-FONTSIZE

Set the fontsize in PROforma points which should be used for the text in infostring objects.

---

defines for PW_TYPE_INFOTEXT

INFOTEXT-INK-COLOUR

Define the RGB colour in which the text in infotext objects has to be displayed.

INFOTEXT-PAPER-COLOUR

Define the RGB paper colour of the infotext objects.

INFOTEXT-FONT

Set the PROforma font which should be used by the infotext objects.

INFOTEXT-FONTSIZE

Set the fontsize in PROforma points which should be used for the text in infotext objects.

---

defines for PW_TYPE_iTEM

ITEM-BORDER-COLOUR

Define the corlour which should be used for the border around the (current) item. The colour is given as an RGB colour, so by specifying the red, green and blue components. This will default to the default middleground colour.

SYSTEM-BACKGROUND-COLOUR

Set the RGB colour which should be used to remove the border around the current item. By default, the global background colour is used when this is not defined explicitly.

ITEM-BORDER-WIDTH

Set the width of the border which should be displayed. The value is given in PROforma coordinates, so with a virtual screen size of 720 by 540.

---

defines for PW_TYPE_LABEL

LABEL-INK-COLOUR

The RGB Colour which has to be used to display the label name. The ProWesS middleground colour will be used as default when not defined.

LABEL-FONT

Set the font to display the label name. The ProWesS default font will be used when not defined.

LABEL-FONTSIZE

Set the fontsize to display the label name. The ProWesS default fontsize will be used when not defined.

---

defines for PW_TYPE_LISTSELECT

LISTSELECT-ARROW-COLOUR

Set the colour (RGB) which should be used for the arrow which is displayed in the listselect item. When not specified, the default middleground colour is used.

LISTSELECT-ARROW-SIZE

Set the size for the arrow which is displayed in the listselect item.

---

defines for PW_TYPE_LOOSE_ITEM

LOOSE-ITEM-AVAILABLE-INK-COLOUR
: Set the RGB colour to display the text in the item when the item is available.

LOOSE-ITEM-UNAVAILABLE-INK-COLOUR
: Set the RGB colour to display the text in the item when the item is unavailable.

LOOSE-ITEM-SELECTED-INK-COLOUR
: Set the RGB colour to display the text in the item when the item is selected.

LOOSE-ITEM-AVAILABLE-PAPER-COLOUR
: Set the RGB colour for the background in an available loose item.

LOOSE-ITEM-UNAVAILABLE-PAPER-COLOUR
: Set the RGB colour for the background in an unavailable loose item.

LOOSE-ITEM-SELECTED-PAPER-COLOUR
: Set the RGB colour for the background in a selected loose item.

LOOSE-ITEM-FONT
: Set the font which has to be used to display the text inside the loose item. If this is not defined, the ProWesS default font is used.

LOOSE-ITEM-FONTSIZE
: Set the fontsize to display the text. When this is not defined, the ProWesS default fontsize is used.

LOOSE-ITEM-AUTOREPEAT-TIMEOUT
: Set the timeout value for the test for autorepeat.

---

defines for PW_TYPE_MENU

MENU-BORDER-WIDTH

Set the width of the border which should be displayed around the current item. The value is given in PROforma coordinates, so with a virtual screen size of 720 by 540.

MENU-AVAILABLE-INK-COLOUR

Set the RGB colour to display the text in the item when the item is available.

MENU-UNAVAILABLE-INK-COLOUR

Set the RGB colour to display the text in the item when the item is unavailable.

MENU-SELECTED-INK-COLOUR

Set the RGB colour to display the text in the item when the item is selected.

MENU-AVAILABLE-PAPER-COLOUR

Set the RGB colour for the background in an available loose item.

MENU-UNAVAILABLE-PAPER-COLOUR

Set the RGB colour for the background in an unavailable loose item.

MENU-SELECTED-PAPER-COLOUR

Set the RGB colour for the background in a selected loose item.

MENU-FONT

Set the font which has to be used to display the text inside the loose item. If this is not defined, the ProWesS default font is used.

MENU-FONTSIZE

Set the fontsize to display the text. When this is not defined, the ProWesS default fontsize is used.

MENU-PAPER-COLOUR

Define the colour which should be used as background colour inside the menu. This will default to the system background colour.

MENU-INK-COLOUR

Set the colour which should be used to display the items inside the menu. This defaults to the system foreground colour.

MENU-BORDER-COLOUR

Set the colour which is used to display the border around the current item. The default value is the system middleground colour.

MENU-DISPLAY-ROWS

Indicates whether the items should be displayed by row or by column. The default is by column. The value is either "true" or "false".

---

defines for PW_TYPE_SCROLL

SCROLL-ARROWS-LEFT
: Make sure that the scroll arrows are always displayed to the left of the scroll bar in horizontal scroll objects.

SCROLL-ARROWS-RIGHT
: Make sure that the scroll arrows are always displayed to the right of the scroll bar in horizontal scroll objects.

SCROLL-ARROWS-ABOVE
: Make sure that the scroll arrows are always displayed above the scroll bar in vertical scroll objects.

SCROLL-ARROWS-BELOW
: Make sure that the scroll arrows are always displayed below the scroll bar in vertical scroll objects.

SCROLL-BAR-MARGINWIDTH
: Specify the marginwidth which is used inside the scroll arrow. This margin is always visible around the bar which indicates the visible area.

SCROLL-ARROW-SIZE
: Define the size of the scroll arrows. The size is given in pt (PROforma coordinates).

SCROLL-ARROW-COLOUR
: Define the colour of the scroll arrows. The colour is given by specifying the RGB components. When not specified, the default ProWesS foreground colour is used.

SCROLL-BAR-BACKGROUND-COLOUR
: Set the RGB colour which is used as background in the scrollbar. If this is not defined, then the ProWesS default background colour is used.

SCROLL-BAR-COLOUR
: Set the RGB colour which is used to display the current size and position of the visible area in the scrollbar. This colour is also used to display the scroll arrows. If this colour is not defined, then the ProWesS default middleground colour is used.

---

defines for PW_TYPE_SEPARATOR

SEPARATOR-COLOUR

Set the colour to be used by separator objects and containers. The parameter is an RGB colour, where each component is specified as a percentage. When this colour is not defined, the ProWesS middleground default colour will be used.

SEPARATOR-THICKNESS

Set the thickness of the separator and container objects. The parameter is given in PROforma coordinates. The thickness is always at least one pixel, even if the parameter was zero.

---

defines for ProWesS system

SYSTEM-SHADOW-RIGHT

Defines the width of the shadow at the right of the window, in pixels.

SYSTEM-SHADOW-BOTTOM

Defines the width of the shadow below the window, in pixels.

SYSTEM-BORDER-WIDTH

Defines the width of the width of the border, in pixels.

SYSTEM-BORDER-COLOUR

Defines the border colour, in device colour.

SYSTEM-SCALEBORDER-WIDTH

When a window can be moved or scaled, then the border is extended with the scaleborder. This is the area which has to be indicated to initiate a move or scale. The scaleborder is surrounded by the normal border. The value is in pixels.

SYSTEM-SCALEBORDER-COLOUR

Colour for the scaleborder, in device colour.

SYSTEM-PREVIEW-MOVE

Should the new position of the window be previewed when moving, 'true' or 'false'.

SYSTEM-PREVIEW-SCALE

Should the new size and position of the window be previewed while scaling, 'true' or 'false'.

SYSTEM-PREVIEW-TIMEOUT

Set the timeout value which should be used when a preview should be given of the window during window move or scale. The default value is 10. The unit is ticks. There are between 50 and 72 ticks a second (depending on your system and country). If the window should be previewed (as set by SYSTEM-PREVIEW-MOVE and SYSTEM-PREVIEW-SCALE), then a preview will be shown at the requested interval.

SYSTEM-LOAD-RESIDENT-FONT

ProWesS types may use PROforma fonts to display text. To make sure that fonts don't have to be reloaded all the time, they can be kept resident by specifying the name as parameter. This becomes essential when the fonts have to be loaded from disk. The fonts will be loaded when ProWesS is started. If they have to be loaded later, errors may occur.

However, the types will probably not report these and just continue.

SYSTEM-FONT-CALCULATED

Make sure that the given font is always available and that the characters are pre calculated at the given size. This allow maximum speed when all the often used combinations are available, as the character never have to be rendered. This will take up some memory though (the pre-calculated glyphs are not stored in the cache to make sure they are never released). You first have to pass the size, and then the fontname, e.g.

```
SYSTEM-FONT-CALCULATED 10 Goudy Old Style
```

SYSTEM-SCROLL-DISTANCE

Set amount which should be scrolled for windows which are bigger than the screen (or bigger than the primary). The parameter should be a value in PROforma coordinates.

For each application, the scrolling distance is limited to at most 3/4 of the window size in that direction.

SYSTEM-DRAGTEST-TIMEOUT

Timout value which should be used for testing whether a hit/do or drag action occurs. If this is too small, then some hit or do events could be interpreted as dragging. If it is too high, then the response to a hit or do may be sluggish.

SYSTEM-BACKGROUND-COLOUR

Set the background colour of the window, given as a device independent RGB colour, where 100 100 100 is white, 0 0 0 is black, 100 0 0 is red, 0 100 0 is green and 0 0 100 is blue. This value is also used by many ProWesS types as default background colour.

SYSTEM-FOREGROUND-COLOUR

Set the default ProWesS foreground colour. This is used by many ProWesS types as default colour. It is typically used to as colour to display important information. The colour is given by specifying the RGB components.

SYSTEM-MIDDLEGROUND-COLOUR

Set the default ProWesS middleground colour. This is used by many ProWesS types as default colour. It is typically used to as colour to display guidelines etc. These thing which are not really important, but are displayed to make the window look better and make the programs easier to use.

SYSTEM-FONT
: Set the font which should be used by default by the ProWesS types (and possibly also by some applications).

SYSTEM-FONTSIZE
: Set the default fontsize which should be used by the ProWesS types (and possibly also by some applications).

PROGS, Professional & Graphical Software
last edited December 27, 1996

defines for PW_TYPE_TITLE_ITEM

TITLE-ITEM-INK-COLOUR

Set the ink colour which is used to display the title in the title item. As normal, the colour is specified by the RGB components. If the ink colour is not specified, the ProWesS default foreground colour is used.

TITLE-ITEM-SURROUND-COLOUR

Set the colour which is used as "border" around the title string. The colour is given by stating the percentages of the RGB components. By default the ProWesS middleground colour is used when no colour is explicitly given.

TITLE-ITEM-PAPER-COLOUR

Set the colour which is used as background under the title string. The ProWesS default background colour is used when no specific value has been assigned.

TITLE-ITEM-FONT

Set the font which should be used for the title name. If no value is given, the ProWesS default font is used.

TITLE-ITEM-FONTSIZE

Set the fontsize (in points) to be used for displaying the title string. If this definition constant is not passed, then the ProWesS default fontsize will be used.

Proposed Entities

HTML references the "Added Latin 1" ENTITY set, which only supplies named entities for a subset of the non-ASCII characters in [ISO-8859-1], namely the accented characters.The following entities are supported by the HTML reader and can only be referenced symbolically.The names for these entities are taken from the appendixes of [SGML].

ENTITY nbsp	" " -- no-break space	
ENTITY iexcl	"¡" -- inverted exclamation mark	¡
ENTITY cent	"¢" -- cent sign	¢
ENTITY pound	"£" -- pound sterling sign	£
ENTITY curren	"¤" -- general currency sign	¤
ENTITY yen	"¥" -- yen sign	¥
ENTITY brvbar	"¦" -- broken (vertical) bar	¢
ENTITY sect	"§" -- section sign	¦
ENTITY uml	"¨" -- umlaut (dieresis)	§
ENTITY copy	"©" -- copyright sign	©
ENTITY ordf	"ª" -- ordinal indicator, feminine	ª
ENTITY laquo	"«" -- angle quotation mark, left	«
ENTITY not	"¬" -- not sign	¬
ENTITY shy	"" -- soft hyphen	
ENTITY reg	"®" -- registered sign	®
ENTITY macr	"¯" -- macron	¯
ENTITY deg	"°" -- degree sign	°
ENTITY plusmn	"±" -- plus-or-minus sign	±
ENTITY sup2	"²" -- superscript two	²
ENTITY sup3	"³" -- superscript three	³
ENTITY acute	"´" -- acute accent	´
ENTITY micro	"µ" -- micro sign	µ
ENTITY para	"¶" -- pilcrow (paragraph sign)	¶

ENTITY middot	"·" -- middle dot	·
ENTITY cedil	"¸" -- cedilla	¸
ENTITY sup1	"¹" -- superscript one	¹
ENTITY ordm	"º" -- ordinal indicator, masculine	º
ENTITY raquo	"»" -- angle quotation mark, right	»
ENTITY frac14	"¼" -- fraction one-quarter	¼
ENTITY frac12	"½" -- fraction one-half	½
ENTITY frac34	"¾" -- fraction three-quarters	¾
ENTITY iquest	"¿" -- inverted question mark	¿
ENTITY Agrave	"À" -- capital A, grave accent	À
ENTITY Aacute	"Á" -- capital A, acute accent	Á
ENTITY Acirc	"Â" -- capital A, circumflex accent	Â
ENTITY Atilde	"Ã" -- capital A, tilde	Ã
ENTITY Auml	"Ä" -- capital A, dieresis or umlaut mark	Ä
ENTITY Aring	"Å" -- capital A, ring	Å
ENTITY AElig	"Æ" -- capital AE diphthong (ligature)	Æ
ENTITY Ccedil	"Ç" -- capital C, cedilla	Ç
ENTITY Egrave	"È" -- capital E, grave accent	È
ENTITY Eacute	"É" -- capital E, acute accent	É
ENTITY Ecirc	"Ê" -- capital E, circumflex accent	Ê
ENTITY Euml	"Ë" -- capital E, dieresis or umlaut mark	Ë
ENTITY Igrave	"Ì" -- capital I, grave accent	Ì
ENTITY Iacute	"Í" -- capital I, acute accent	Í
ENTITY Icirc	"Î" -- capital I, circumflex accent	Î
ENTITY Iuml	"Ï" -- capital I, dieresis or umlaut mark	Ï
ENTITY ETH	"Ð" -- capital Eth, Icelandic	Ð
ENTITY Ntilde	"Ñ" -- capital N, tilde	Ñ
ENTITY Ograve	"Ò" -- capital O, grave accent	Ò
ENTITY Oacute	"Ó" -- capital O, acute accent	Ó
ENTITY Ocirc	"Ô" -- capital O, circumflex accent	Ô
ENTITY Otilde	"Õ" -- capital O, tilde	Õ
ENTITY Ouml	"Ö" -- capital O, dieresis or umlaut mark	Ö

ENTITY times "×" -- multiply sign ×
ENTITY Oslash "Ø" -- capital O, slash Ø
ENTITY Ugrave "Ù" -- capital U, grave accent Ù
ENTITY Uacute "Ú" -- capital U, acute accent Ú
ENTITY Ucirc "Û" -- capital U, circumflex accent Û
ENTITY Uuml "Ü" -- capital U, dieresis or umlaut mark Ü
ENTITY Yacute "Ý" -- capital Y, acute accent Ý
ENTITY THORN "Þ" -- capital THORN, Icelandic Þ
ENTITY szlig "ß" -- small sharp s, German (sz ligature) ß
ENTITY agrave "à" -- small a, grave accent à
ENTITY aacute "á" -- small a, acute accent á
ENTITY acirc "â" -- small a, circumflex accent â
ENTITY atilde "ã" -- small a, tilde ã
ENTITY auml "ä" -- small a, dieresis or umlaut mark ä
ENTITY aring "å" -- small a, ring å
ENTITY aelig "æ" -- small ae diphthong (ligature) æ
ENTITY ccedil "ç" -- small c, cedilla ç
ENTITY egrave "è" -- small e, grave accent è
ENTITY eacute "é" -- small e, acute accent é:
ENTITY ecirc "ê" -- small e, circumflex accent ê
ENTITY euml "ë" -- small e, dieresis or umlaut mark ë
ENTITY igrave "ì" -- small i, grave accent ì
ENTITY iacute "í" -- small i, acute accent í
ENTITY icirc "î" -- small i, circumflex accent î
ENTITY iuml "ï" -- small i, dieresis or umlaut mark ï
ENTITY eth "ð" -- small eth, Icelandic ð
ENTITY ntilde "ñ" -- small n, tilde ñ
ENTITY ograve "ò" -- small o, grave accent ò
ENTITY oacute "ó" -- small o, acute accent ó
ENTITY ocirc "ô" -- small o, circumflex accent ô
ENTITY otilde "õ" -- small o, tilde õ
ENTITY ouml "ö" -- small o, dieresis or umlaut mark ö

ENTITY divide	"÷" -- divide sign	÷
ENTITY oslash	"ø" -- small o, slash	ø
ENTITY ugrave	"ù" -- small u, grave accent	ù
ENTITY uacute	"ú" -- small u, acute accent	ú
ENTITY ucirc	"û" -- small u, circumflex accent	û
ENTITY uuml	"ü" -- small u, dieresis or umlaut mark	ü
ENTITY yacute	"ý" -- small y, acute accent	ý
ENTITY thorn	"þ" -- small thorn, Icelandic	þ
ENTITY yuml	"ÿ" -- small y, dieresis or umlaut mark	ÿ

This file is extracted from the HTML 2.0 specification