werc 指南

极简反框架与 *Plan 9* 哲学的网络回响

"Werc is a minimalist web anti-framework built following the Unix and Plan 9 tool philosophy *of software design."*
— werc.cat-v.org

"Database free, uses files and directories instead."
— werc 第一原则

"When people ask me what I look for in a woman, I'm going to say, 'werc-like qualities'."
— shardz

An Anti-Framework

一个反对 Web 框架繁文缛节的网络系统，
高度模块化的核心仅约 150 行 `rc` 脚本，
不依赖数据库，只依赖文件与目录。

蒸馏自 `werc.cat-v.org` 及其文档、应用、Wiki

2026 年 5 月 11 日

目录

前言：为什么是 **werc**

> *"Minimize tedious work: eg., no need to ever write HTML, use markdown (or any other format) instead."*
> — werc 简介

当代 Web 框架是繁殖学的奇迹：每一个新版本都生出新的依赖、新的脚手架、新的 YAML、新的迁移脚本，直到没有人记得最初要做的不过是把一些文字放上互联网。`cat-v.org` 圈子对此的答复异常简洁：不要框架，要工具。**werc** 就是这一答复的网站层。

werc 不是一个 CMS，也不只是一个静态站点生成器。它是一个*反框架*（anti-framework）：

- 数据存在文件与目录里，而不是数据库里；
- 核心逻辑约 150 行 `rc` shell 脚本，剩下的能力以可插拔的 "apps" 形式存在；
- 你可以用 `mkdir`、`echo`、自己最喜欢的编辑器来管理一个站点；
- 一个安装可托管任意数量的虚拟域名，每个站点继承一组共同的样式与布局，并可以在全局、域名族、单一域名、目录子树、单个文件五种层级上独立定制。

本指南把 `werc.cat-v.org` 上散落的文档、Wiki、应用说明、路线图汇集为一本可以从头读到尾的小书。我们假设读者已经对 Unix shell、HTTP、Markdown 有基本概念，但对 `rc` 与 Plan 9 完全陌生也无妨。

> **阅读建议。**如果你只想尽快把 werc 跑起来，直接读*第 2 章*和*第 3 章*中你所用 Web 服务器对应的那一节；之后再回到第 1 章理解 "为何如此"，最后用第 4–10 章按需查阅。

Chapter 1

哲学与缘起

"Very minimalist yet extensible codebase: highly functional core is 150 lines, with extra functionality in modular apps."
— werc.cat-v.org

1.1 反框架（Anti-Framework）

werc 自我定位为一个"反 Web 框架"（web anti-framework）。它的存在不是为了与 Django、Rails、Next.js 在同一个赛道上竞争，而是要从根本上拒绝它们对世界的描述：把"写一些可以被浏览器读到的文本"这件事，重新还原为"写一些文件"。

> *Werc avoids the pain of managing collections of websites and developing web applications.*

它的设计哲学直接来自 Unix 与 Plan 9 的工具哲学（tool philosophy）：

- 无数据库。内容、配置、用户、权限——全部是文件和目录。可以用 `cp`、`mv`、`rsync`、`git` 管理。
- 用 **`rc`** 写。核心调用一组标准的 Unix / Plan 9 命令；不需要"运行时"，不需要"虚拟机"，只需要一个能跑 `rc` 与 CGI 的 HTTP 服务器。
- 不替你写 **HTML**。站点写 Markdown（或者你喜欢的任何格式）；werc 会把它套进站点共同的模板，输出 HTML 5。
- 可拼装。给博客加评论？启用 `bridge` app。把一个文档目录变成可编辑的 wiki？启用 `dirdir` app。App 之间的组合在 `_werc/config` 里一行函数调用就能完成。
- 自由扩展。werc 的"app"可以用任何语言写，只要它能在 CGI 环境里输出 HTML 片段。官方建议依然是 `rc`。

1.2 这意味着什么

把以上几条放在一起观察，会得到一种相当奇异的产品形态：

- 站点等于 `sites/<domain>/` 下的目录树。要发布一篇文章，在合适的位置 `echo`/`cat` 出一个 `.md` 文件即可。
- 配置等于该目录树中散布的 `_werc/config` 文件。每个目录都可以重写父目录的行为。
- 用户与组等于 `etc/users/` 下的目录。一个用户是一个目录，目录里放 `password` 文件；一个组是同样位置的目录，里面放 `members` 文件。
- 部署等于把整个目录树 `rsync` 到任意一台能跑 CGI 的机器上。

> 这种“一切都是目录”的设计与 Plan 9“一切都是文件”的设计同源。熟悉 9P、`rc`、`acme` 的人会很快感到亲切；其他人需要适应一段时间，但适应之后会发现“运维网站”与“写一些目录”之间已经没有缝隙。

1.3　特性一览

下面是 werc 在自家首页列出的一些卖点，供后续章节作参照：

- 与已有内容融洽相处：你可以把 HTML 或纯文本文件直接放进站点，它们会无缝融入；
- 设计为同时托管多个虚拟域名，它们共享同一套样式和布局；
- 配置可在以下层级上覆盖：全局 / 域名族（domain group）/ 单域名 / 目录子树 / 单文件；
- 同一台机器可以并排跑多个定制版的 werc；
- 用户管理与权限系统简单且足够灵活；
- App 可以组合：例如博客 + 评论、文档树 + wiki；
- 可以用任何语言写 app（推荐 `rc`）。

1.4　运行环境

werc 只需要：

- 一些 Plan 9 命令：`cat`、`grep`、`sed`、`rc` 等；
- 一个支持 CGI 的 HTTP 服务器。

可用的 Plan 9 用户态有三种来源：

- **Plan 9 from User Space**（也叫 `plan9port` / `p9p`）：功能完整，适合 Linux、*BSD、macOS、Solaris；
- **9base**（来自 suckless.org）：最小化版本，适合资源紧张或追求极简的环境；
- **frontbase / frontport**（来自 9front）：另一支由 9front 维护的端口。

HTTP 服务器没有限制，已知能跑起来的至少包括 Apache、Lighttpd、Cherokee、nhttpd、Hiawatha、rc-httpd、cgd、nginx（通过 fcgiwrap）等。在 Plan 9 上则可以直接用原生 httpd。

1.5 许可与归属

werc 在 `cat-v.org` 体系中属于“公有领域”（public domain）作品，理由典型地是“所谓的‘知识产权’是一种自相矛盾的提法”。如果你的国家承认不了公有领域，可以按 MIT 与 ISC 协议使用。

致谢按官方页面的顺序：

- **Uriel**，werc 的最初作者；
- **Kris Maglione（aka JG）**，实现了 RSS、awk 的 rc 模板系统等；部分代码源于他的 diri wiki；
- **Mechiel（aka oksel）**，写了 `md_cache`；
- **Garbeam（aka arg）**，最初的 diri 代码作者，证明了“用 `rc` 写复杂 Web 应用”是可行的；
- **Ethan Gardner**，rc-httpd 作者。

Chapter 2

获取与最简部署

“A beautiful thing about werc is that it deals solely with flat files and directories.”
— Debian + Lighttpd 快速安装

2.1 获取源码

werc 的源码托管在 9front 维护的 Mercurial 仓库里：

```
hg clone https://code.9front.org/hg/werc/
```

仓库可在线浏览：https://code.9front.org/hg/werc/。近年的开发也镜像到了 git://shithub.

2.2 目录骨架

解包以后，整个发行版大致是这样：

- `bin/werc.rc` ——入口 CGI 脚本，HTTP 服务器把请求转发给它；
- `bin/aux/` ——辅助脚本（如 `addwuser.rc`、`bpst.rc`）；
- `etc/initrc` ——全局初始化与默认配置；
- `etc/users/` ——用户与组目录；
- `lib/` ——默认模板与片段（`.tpl`、`.inc`）；
- `tpl/` ——系统级模板，如 `sitemap.tpl`、`404.tpl`；
- `pub/` ——站点公共静态资源，如默认 CSS 与图标；
- `apps/` ——内置应用（`blagh`、`bridge`、`dirdir`、`wman`、`duckduckgo` 等）；
- `sites/` ——每个目录对应一个虚拟主机；这是大多数日常编辑发生的地方。

2.3 Debian + Lighttpd 五分钟安装

下面这段安装手册来自官方的 *Werc Quick Setup for Debian Linux+Lighttpd*，原作者基于 Debian 5.0.7 的 netinstall。直到今天，它依然是理解 “werc 是怎么跑起来的” 的最直接路径。

```
# 1. 装好 lighttpd 与 build-essential（make 是构建 9base 的 mk 所必需
    的）
aptitude install lighttpd build-essential
curl http://dl.suckless.org/tools/9base-6.tar.gz | tar xzf -
curl http://hg.cat-v.org/werc/archive/tip.tar.bz2 | tar xjf -
```

```
# 2. 编译 9base（提供 mk 与其他 Plan 9 命令）
cd 9base-6
make install clean
```

```
# 3. 把 werc 放到合适的位置，并复制一个示例站点
mv ../werc-bec1802070f8/ /var/www/werc
cp -r /var/www/werc/sites/default.cat-v.org \
      /var/www/werc/sites/your.domain.com
echo '# Hello World!' > \
     /var/www/werc/sites/your.domain.com/index.md
```

```
# 4. 配置 lighttpd（核心是把 404 转给 werc.rc，启用 .rc CGI）
$HTTP["host"] =~ "^your\.domain\.com$" {
    index-file.names = ( )
    server.error-handler-404 = "/werc.rc"
    alias.url    += ( "/werc.rc" => "/var/www/werc/bin/werc.rc" )
    cgi.assign   += ( ".rc" => "" )
    server.dir-listing = "disable"
}
```

到这里访问 `your.domain.com` 就能看到 “Hello World”。往这个 `index.md` 旁边新增任何 `.md` 文件、任何子目录，它们都会立即成为站点的一部分，无需重启服务。

> 开发须知。官方明确提示 “操作系统会变，例子会过时”。本节展示的是 “形状” 而非每条命令的当代发行版包名。真正部署时请按你当时的发行版与 werc 版本对照修改。

2.4 OpenBSD 与其它平台

官方 quick-setup 目录还提供了 *OpenBSD + Apache 1.3* 的示例。在 9front / Plan 9 上 werc 与 `rc-httpd` 协作得最自然（参见第 3 章对应小节）。macOS 上用 plan9port 也能跑，但通常只用作开发环境。

2.5　静态链接 Plan 9 from User Space

在类 Unix 系统上，动态链接对 `fork` 调用造成的额外开销不小，而 werc 几乎每次请求都会派生若干个 Plan 9 命令。官方 *Tips & Tricks* 给出的优化是把 *p9p* 静态链接：

```
# 编辑 $PLAN9/bin/9l
diff -r 47b3d93f532d bin/9l
@@ -264,7 +264,7 @@
        extralibs="$extralibs -lutil"
        ;;
 *Linux*)
-       ld=gcc
+       ld="gcc -static"
        userpath=true
        extralibs="$extralibs -lutil"
```

随后重新执行 p9p 的 INSTALL 脚本即可。在中等流量下，这个改动通常能把单次请求的尾延迟从几十毫秒压到十几毫秒以内。

Chapter 3

HTTP 服务器接入

"Werc can use any HTTP server that can handle CGI."

— werc.cat-v.org

werc 与 HTTP 服务器之间的接口非常简单：

- 把 `.rc` 文件作为 CGI 脚本运行；
- 把找不到的路径（404）交给 `/werc.rc` 处理；
- 关闭服务器自身的目录列表（让 werc 来生成）；
- 可选：用 alias / rewrite 让服务器自己处理 `pub/`、`favicon.ico` 等静态资源。

下面给出几种最常见的接入配置。每段都可以独立阅读。

3.1 lighttpd

需要启用 `mod_cgi`、`mod_alias`，以及（非极简部署）`mod_rewrite`；`mod_setenv` 在需要调整环境变量（特别是 `PATH`）时有用。

3.1.1 单域最小配置

```
$HTTP["host"] =~ "^test\.cat-v\.org$" {
    index-file.names = ( )
    server.error-handler-404 = "/werc.rc"
    alias.url    += ( "/werc.rc" => "/var/www/cat-v.org/bin/werc.rc" )
    cgi.assign   += ( ".rc" => "" )
    server.dir-listing = "disable"
}
```

3.1.2 多域 + 静态资源 + 外挂 CGI

下面这段是 cat-v.org 自家真实配置的精简版，演示了"由 lighttpd 自己服务静态文件 `pub/`、`favicon.ico`，并挂载额外的 hg CGI"：

```
$HTTP["host"] =~ "^((harmful|9p|gsoc|doc|uriel|src|repo|www|)(\.|)cat-v
   \.org|(www\.)?binarydream.org|)$" {
    index-file.names = ( )
    evhost.path-pattern      = "/var/www/cat-v.org/sites/%3.%0/"
    server.error-handler-404 = "/werc.rc"

    alias.url   += ( "/pub/"        => "/var/www/cat-v.org/pub/" )
    alias.url   += ( "/favicon.ico" => "/var/www/cat-v.org/pub/
       default_favicon.ico" )
    alias.url   += ( "/doc/"        => "/var/www/cat-v.org/sites/doc.cat
       -v.org/" )
    alias.url   += ( "/werc.rc"     => "/var/www/cat-v.org/bin/werc.rc"
       )
    alias.url   += ( "/debug.rc"    => "/var/www/cat-v.org/bin/debug.rc"
        )
    cgi.assign  += ( ".rc"  => "" )
    cgi.assign  += ( ".cgi" => "" )
    server.dir-listing = "disable"

    url.rewrite-once = ( "/hg/(.*)" => "/hg/hgwebdir.cgi/$1" )
    alias.url   += ( "/hg/" => "/var/www/cat-v.org/bin/" )
}
```

3.2 Apache 2

Apache 没有原生的 evhost 风格虚拟主机映射，但 RewriteRule 足够干净地完成同样的工作：

```
<VirtualHost *:80>
    RewriteEngine On
    PassEnv PLAN9          # 若 p9p 不在 /usr/local/plan9

    ServerName cat-v.org
    ServerAlias www.cat-v.org harmful.cat-v.org *.cat-v.org

    AddHandler cgi-script .rc

    <Directory /srv/werc/bin>
        Options ExecCGI
        AllowOverride None
        Require all granted
    </Directory>
    <Directory /srv/werc/sites>
        Options FollowSymLinks
```

```
        AllowOverride None
        Require all granted
    </Directory>

    <IfModule mod_dir.c>
        DirectoryIndex /werc.rc
    </IfModule>

    RewriteRule /pub/style/style.css /srv/werc/pub/style/style.css
    RewriteRule /favicon.ico         /srv/werc/pub/favicon.ico
    RewriteRule (.*)                 /srv/werc/sites/%{HTTP_HOST}/$1

    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule .* /srv/werc/bin/werc.rc

    DocumentRoot "/srv/werc/bin/"
    ErrorDocument 404 /werc.rc
</VirtualHost>
```

原始示例使用 Apache 2.2 风格的 `Order allow,deny`；在 Apache 2.4 上请改写为 `Require all granted`（如上）。

3.3 nginx

nginx 没有原生 CGI，所以 werc 在 nginx 下要走 FastCGI：通常组合是 `spawn-fcgi` + `fcgiwrap`（或 `cgd`）。下面是 cat-v.org 文档里给出的最小可工作配置（精简版）：

```
server {
    listen 80;
    server_name test.cat-v.org;

    location / {
        fastcgi_pass localhost:9000;
        fastcgi_param  QUERY_STRING       $query_string;
        fastcgi_param  REQUEST_METHOD     $request_method;
        fastcgi_param  CONTENT_TYPE       $content_type;
        fastcgi_param  CONTENT_LENGTH     $content_length;
        fastcgi_param  SCRIPT_NAME        /var/www/werc/bin/werc.rc;
        fastcgi_param  REQUEST_URI        $request_uri;
        fastcgi_param  DOCUMENT_URI       $document_uri;
        fastcgi_param  DOCUMENT_ROOT      $document_root;
        fastcgi_param  SERVER_PROTOCOL    $server_protocol;
        fastcgi_param  GATEWAY_INTERFACE  CGI/1.1;
        fastcgi_param  SERVER_SOFTWARE    nginx/$nginx_version;
```

```
        fastcgi_param  REMOTE_ADDR        $remote_addr;
        fastcgi_param  REMOTE_PORT        $remote_port;
        fastcgi_param  SERVER_ADDR        $server_addr;
        fastcgi_param  SERVER_PORT        $server_port;
        fastcgi_param  SERVER_NAME        $server_name;
        fastcgi_param  REMOTE_USER        $remote_user;
    }
}
```

实际生产配置通常会另开 `location /pub/` 等让 nginx 直接服务静态文件。

3.4 rc-httpd

`rc-httpd` 是 Plan 9 / 9front 上的轻量 HTTP 服务器，最新的 werc 发行版直接在 `bin/contrib/rc-httpd/` 下附带了它。最小接入方式是在 `select-handler` 中加：

```
if(~ $SERVER_NAME *){
    PATH_INFO=$location
    FS_ROOT=/path/to/werc/sites/$SERVER_NAME
    exec static-or-cgi /path/to/werc/bin/werc.rc
}
if not
    error 503
```

9front 默认就装有 `rc-httpd`（在 `/rc/bin/rc-httpd/`）；其它 Plan 9 系发行版按照上面配置同样可用。

3.5 Dreamhost：在受限的共享主机里活下来

Dreamhost 不允许租户改 `httpd.conf`，但 `.htaccess` 允许有限的指令。社区给出的部署配方如下：

```
# 1. 把 plan9port 或 9base 装到家目录
#    /home/username/plan9

# 2. 把 werc 解到域名目录
#    /home/username/domain.com
#    /home/username/domain.com/werc/sites/domain.com/

# 3. 把 etc/initrc 复制为 initrc.local 并修改：
plan9port=/home/username/plan9

# 4. 修改 werc.rc 的解释器：
#!/home/username/plan9/bin/rc
```

```
# 5. 新建 werc.cgi（共享主机通常要求 .cgi 后缀走 CGI）
#!/bin/sh
export PLAN9=/home/username/plan9
export PATH=$PLAN9/bin:$PATH
exec ./werc.rc

# 6. 在域名根目录放 .htaccess：
ReWriteEngine On
<IfModule mod_dir.c>
  DirectoryIndex /werc.cgi
</IfModule>

RewriteRule (.*) /home/username/domain.com/werc/sites/%{HTTP_HOST}/$1
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule .* /home/username/domain.com/werc/bin/werc.cgi

RewriteRule /werc.rc /home/username/domain.com/werc/bin/werc.cgi
ErrorDocument 404 /werc.cgi
```

这个配方的意义不在于 Dreamhost 本身，而在于它示范了“如何把 werc 塞进任何只允许 `.htaccess` 的环境”。

Chapter 4

站点布局与基本工作流

"All you need is mkdir and a text editor of your choice."

— werc FAQ

4.1 一个站点等于一个目录

werc 中所有公开内容都活在 `sites/<domain>/` 下面。例如，假设你的安装位于 `/var/www/werc`：

```
/var/www/werc/sites/
    cat-v.org/
        index.md
        _werc/
            config
            lib/
                footer.inc
        about/
            index.md
            people.md
    blog.cat-v.org/
        _werc/config
        2024/
            03/
                14/
                    why-plan9.md
```

理解 werc 只需要记住几条规则：

- 站点对应的目录就是 HTTP 路径的根。访问 `/about/people` 时，werc 会去找 `about/people.md`、`about/people.txt`、`about/people.html` 或 `about/people/index.{md,html,txt}`。
- 没有 index 文件时回退到目录列表（详见下文）。
- `_werc/` 是该目录及其子树的“配置面板”；它对外不可见，但里面任何 `config`、`lib/`、`pub/` 都会影响这个子树的行为。

4.2 Markdown 与其它格式

默认的格式化命令是：

```
formatter=(fltr_cache md2html.awk)
```

它将 `.md` 文件喂给 yiyus 的纯 awk Markdown 实现，并把结果缓存。你也可以把 `formatter` 改成任何能从命令行读文件或 stdin 并输出 HTML 的程序，例如：

```
# 改用 discount
formatter=(fltr_cache markdown)
```

werc 同样能直接服务：

- 已有的 `.html` 文件（直接套用模板）；
- 纯 `.txt`（用 `<pre>` 包裹）；
- 任何带主处理器（*main handler*）的目录（如 dirdir、blagh、wman 把目录变成 wiki / 博客 / man 页浏览器）。

4.3 目录列表与 _header / _footer

如果一个目录里既没有 `index.md`、`index.html`、`index.txt`，也没有任何 app 注册的“主处理器”，默认目录处理器会负责生成列表页：列出当前目录路径，加上该目录下所有文件与子目录的链接。

你可以选择性地在目录里放：

- `_header.md`：列表之前显示；
- `_footer.md`：列表之后显示。

排序由配置变量 `dir_listing_ls_opts` 控制，它接受 Plan 9 `ls(1)` 的选项。例如按时间倒序排列：

```
dir_listing_ls_opts=( -t )
```

4.4 站点分组与 masterSite

一个 werc 安装常常承载一族相关域名。“站点分组”（site group）让一族站点共享模板、样式、配置。做法是在子站点的 `_werc/conf` 中声明继承自哪个“主站”：

```
# sites/foo.bar.com/_werc/conf
masterSite=bar.com
```

设置之后，对模板（如 `404.tpl`）或 `.inc` 文件的查找顺序变为：

1. `sites/foo.bar.com/_werc/lib/404.tpl`

2. `sites/bar.com/_werc/lib/404.tpl`
3. 缺省的 `lib/404.tpl`

这套查找规则让你能在一个目录里集中维护一族站点的视觉与文案，而单一站点仍可以越级覆盖。

4.5　文件类型速查

`.md` Markdown 内容文件，由 `formatter` 处理。

`.html` 原样 HTML，会被嵌入到模板里。

`.txt` 纯文本，被 `<pre>` 包裹。

`.tpl` `rc` 模板文件（见第 6 章）。

`.inc` 纯文本片段，无任何处理，原样插入。

Chapter 5

配置：在每一层覆盖

"Configuration and customization can be at any level: global, per-domain-group, domain-wide, directory sub-tree, and single file."
— werc.cat-v.org

werc 的配置不集中在一个"大 YAML"中，而是沿着目录树分布：

- 全局：`etc/initrc`、`etc/initrc.local`；
- 域名族：主站的 `_werc/config`；
- 单域名：该域名 `sites/<domain>/_werc/config`；
- 子目录：该目录下的 `_werc/config`；
- 单个文件：模板与 inc 的覆盖式查找。

配置文件是真正的 `rc` 脚本，可以设变量、调用函数、定义自己的处理器，甚至直接生成内容。

5.1 基本变量

5.1.1 格式化器

- `formatter` ——文档格式化命令，默认 `(fltr_cache md2html.awk)`。

5.1.2 页面标题与 meta

- `siteTitle`、`siteSubTitle` ——页眉显示的站点标题；
- `meta_description` ——写入 `<meta name="description">`；
- `meta_keywords` ——写入 `<meta name="keywords">`；
- `extraHeaders` ——原样插入 `<head>` 里的 HTML。

5.1.3 侧边栏与导航

- `sideBarNavTitle` ——侧边栏顶部的标题文字；

- 函数 `conf_hide_paths [pattern...]` ——把若干路径模式从导航、sitemap、目录列表里隐藏。警告：它只是“不显示”，不是“访问控制”；不要把它当安全机制用。

5.1.4 HTTP 头与重定向

- `extraHttpHeaders` ——加入响应头的原始 HTTP header；
- 函数 `conf_perm_redirect [pattern] destination` ——若给了 pattern，则对请求 URL 做替换；可以理解为 `s/pattern/destination/`。

5.2 访问控制

权限系统本身不在配置里，而是放到一段 `switch` 模式匹配里。经典写法是：

```
switch ($req_path) {
case /_users/login
case /pub/*
case /robots.txt
case *
    if(! check_user editors)
        perm_redirect /_users/login
}
```

它的含义是：除登录页、公共资源、`robots.txt` 之外的所有路径，若当前请求者不属于 `editors` 组，重定向到登录页。

> `switch` 的“无 `break`、case 串联”行为来自 `rc` 而非 `sh`：没有匹配到的 case 不会落到下一个 case；多个连续的 case 一行一个，仅最后一个 case 的代码块对它们都生效。

5.3 用 initrc.local 加扩展

如果你的处理器是一次性的、只为本机服务，习惯上写在 `etc/initrc.local` 中：它会被 `etc/initrc` 在最后 source 一次，因此可以无侵入地覆盖默认变量与函数。任何“可复用、可能值得分享给其他人”的逻辑则应放到独立的 app 中（见第 9 章）。

Chapter 6

模板与样式

"Lines starting with % are executed as rc commands, the resulting output is inserted in the document."
— The Rc Template Language

6.1 rc 模板语言

werc 的模板语言（由 Kris 实现）几乎只有一条规则："行首是%，剩下交给 `rc`"。

6.1.1 语法

- 以 `%` 开头的行作为 `rc` 命令执行，输出会被插入到当前位置；
- `%{` 与 `%}` 包裹多行 `rc` 代码（注意 `%` 与 `{`、`}` 之间没有空格）；
- `%($my_var%)` 在文本流中内联一个环境变量的值。

6.1.2 例子：三种等价写法

```
<ul>
% for(i in a b c) {
%   echo '<li>'$i'</li>'
% }
</uL>
```

```
<ul>
%{
for(i in a b c) {
    echo '<li>'$i'</li>'
}
%}
</uL>
```

```
<ul>
% for(i in a b c) {
<li>%($i%)</li>
% }
</uL>
```

三段都会产出：

```
<ul>
<li>a</li>
<li>b</li>
<li>c</li>
</ul>
```

要进一步了解 rc 本身，推荐两份原始材料：

- Plan 9 的 *rc(1)* man page：http://man.cat-v.org/plan_9/1/rc；
- Tom Duff 的论文 *Rc, A Shell for Plan 9*：http://rc.cat-v.org.

6.2 模板与 include 文件

6.2.1 放在哪儿

要修改 lib/ 下任何默认模板或 include 文件，直接把它复制到站点的 _werc/lib/ 再编辑即可。要修改 tpl/ 下的系统级模板（如 sitemap.tpl），则把它复制到站点根（如 sites/foo.bar.com/sitemap.tpl）再编辑。

6.2.2 常见 include

top_bar.inc 页面顶部的细窄横条，通常放跨站点链接、口号或留空。

footer.inc 页脚，默认包含登录链接。

headers.inc 原样插入到 <head> 的 HTML。

6.2.3 常见 .tpl

default_master.tpl 主模板，串起侧栏、内容主区、其它处理器。除非要改整体布局，否则一般不动它。

headers.tpl 默认 <head> 内容的模板。

sitemap.tpl 站点地图页模板。

404.tpl 404 页面模板。

6.3 CSS 与配色

要追加站点自己的 CSS，新建一个文件即可：

```
sites/<domain>/_werc/pub/style.css
```

它会在标准 werc 样式表之后自动加载——也就是“追加”而非“替换”。你也可以复制 `pub/style/style.css` 完全自己来定主题。官方给出的最小配色覆盖示例：

```
header nav      { background-color: rgb(100,135,220); color: white; }
header h1       { background-color: #ff6d06; color: black; }
body > nav > div { border-bottom: 1px solid #ddd; }
body > nav > div a       { color: rgb(0, 102, 204); }
body > nav > div a:hover { color: white;
                           background-color: rgb(100,135,220); }
```

6.4 案例：suckless.org 的横向菜单

`suckless.org` 不喜欢 werc 默认的左侧栏（占 16em，移动端不友好），转而采用横向菜单。整体改动分三块：

6.4.1 lib/default_master.tpl

```
<div id="container">
    <div id="header">
        <div class="midHeader">
% if(! ~ $#siteImage 0) {
            <h1 class="headerTitle"><a href="/">
              <img src="%($"siteImage%)" alt="%($"siteTitle%)"/>
              <span id="headerSubTitle">%($"siteSubTitle%)</span>
            </a></h1>
% }
% if not {
            <h1 class="headerTitle"><a href="/">
              %($"siteTitle%)
              <span id="headerSubTitle">%($"siteSubTitle%)</span>
            </a></h1>
% }
        </div>
        <div id="menu">
            <ul>
% nav_sites
            </ul>
        </div>
    </div>
```

```
% if(! ~ $#handlers_bar_left 0) {
%   for(h in $handlers_bar_left) { run_handler $$h }
% }
    <div id="main-copy">
% run_handlers $handlers_body_head
% run_handler  $handler_body_main
% run_handlers $handlers_body_foot
    </div>
    <div id="footer">
% cat `{ get_lib_file footer.inc }
    </div>
</div>
```

6.4.2 etc/initrc.local

```
sitesdir='/var/www/sites'
formatter=(fltr_cache markdown)
sites_menu=(home/:suckless.org code:hg.suckless.org \
            download:dl.suckless.org man/:man.suckless.org \
            dwm/:dwm.suckless.org libs/:libs.suckless.org \
            st:st.suckless.org surf/:surf.suckless.org \
            tools/:tools.suckless.org wmii/:wmii.suckless.org \
            wmi/:wmi.suckless.org)
debug=()

fn nav_sites {
    for(m in $sites_menu) {
        ifs=(':') { co=`{echo -n $m} }
        if(~ 'http://'^$co(2) $base_url)
            echo '<li class="thisPage"><a href="http://'^$co(2)^'/">'^
               $co(1)^'</a></li>'
        if not
            echo '<li><a href="http://'^$co(2)^'/">'^$co(1)^'</a></li>'
    }
}
```

6.4.3 另一种横向菜单实现

如果你想要更接近默认 werc 风格的横菜单，社区有另一种方案：为 “subheader” 注册处理器，输出嵌套的 `<ul class="headerMenu...">`（嵌套层数对应 CSS 类名追加下划线）：

```
handler_subheader='nav_header'

fn nav_header {
```

```
    level=''
    for (d in $req_paths_list) {
    echo '<ul class="headerMenu'$level'">'
    level=$level^_
    ls -F $sitedir/./$d >[2]/dev/null \
       | { sed $dirfilter'/\/[^_.\/][^\/]*(\.(md|txt|html)|\/)$/!d;
                           s!^'$sitedir'!!; '$dirclean
            if(! ~ $#synth_paths 0) echo $synth_paths | tr ' ' $NEW_LINE
       } | sort -u | awk -F/ '
    { d = ""
      if(match($0, "/$")) d = "/"
      sub("/$", "")
      bname = $NF d
      path  = $0 d
      gsub(/[\-_]/, " ", bname)
      pa = path
      gsub(/[^\/]$/, "&/", pa)
      if(index(ENVIRON["req_path"] "/", pa) == 1)
          print "<li><a href=\"" path "\" class=\"thisPage\">" bname "</
              a></li>"
      else
          print "<li><a href=\"" path "\">" bname "</a></li>"
    }
    END { print "</ul>" }'
    }
}
```

要让它生效，记得清空默认左栏：`handlers_bar_left=()`，并补上一段 CSS 把 `.subHeader ul` 排成水平。

Chapter 7

用户、组与权限

"Users and groups share the same namespace."

— User and Group Management

7.1 用户与组就是目录

用户与组信息都存在 `etc/users/` 下：

- 一个用户是一个目录，里面至少有一个文件 `password`，内容就是该用户的密码；
- 一个组是同样位置的一个目录，里面有一个文件 `members`，每行一个成员名；
- 一个目录同时含 `password` 和 `members`，则它既是用户又是组；
- 组 `admin` 是内建的，所有该组成员默认拥有大部分 werc app 的管理员权限。

例：

```
% ls etc/users/
eekee  uriel  yosyp
% cat etc/users/uriel/password
mypass
```

7.2 建立一个用户

最朴素的做法：

```
% mkdir etc/users/glenda/
% echo carrot > etc/users/glenda/password
% mkdir -p   etc/users/rabbits
% echo glenda >> etc/users/rabbits/members
```

更省事的写法：使用 `bin/aux/addwuser.rc`：

```
addwuser.rc user_name user_pass [groups ...]
```

7.3 把权限挂到路径上

权限本身的检查由 `check_user` 与 `perm_redirect` 在配置里完成，如第 5 章“访问控制”小节那段 `switch`。基本套路：

- 在站点根的 `_werc/config` 里写一组 `switch ($req_path) { …}`；
- 白名单出公开路径（登录页、静态资源等）；
- 其余路径 `check_user` 不通过则 `perm_redirect /_users/login`。

由于用户/组只是目录，备份、迁移、审计权限只是 `ls+diff` 的工作。

Chapter 8

Apps：组合代替继承

"Applications can be easily combined."

— werc.cat-v.org

werc 的功能扩展机制叫 **apps**。一个 app 通常包括：

- `apps/<name>/app.rc` ——主处理器与配置入口；
- `apps/<name>/*.tpl`、`*.inc` ——该 app 需要的模板与片段；
- 一个或多个形如 `conf_enable_<name>` 的开启函数，约定俗成用在 `_werc/config` 里。

启用一个 app 的统一姿势是在配置文件里调用对应函数，比如 `conf_enable_blog`、`conf_enable_com`

8.1 blagh ——给反对博客的人用的博客引擎

blagh（昵称 Blag / Blah / Bragg）是 werc 的博客 app，特性：

- 基于文件，无数据库；
- 历史浏览；
- Atom、JSON、RSS 输出；
- 服务器端 feed "聚合"——把多个博客合并为一个 feed；
- 通过 `bridge` 提供评论；
- Markdown 格式。

8.1.1 发帖

两种发帖方式：

- 网页表单（需要登录且属于 `$conf_blog_editors`）；
- 命令行 `bin/aux/bpst.rc`：
  - `-f file` 指定要发布的内容文件，省略则打开 `$EDITOR`；

- 完成编辑后调用 `ispell` 拼写检查（建议装上 ispell）；
- 在博客根目录下生成一个以当前年份命名的隐藏目录结构，你可以先 `ls` 检查再 `mv` 去掉前缀点使其可见。

8.1.2 配置

放在 `_werc/config` 中：

```
# 在本目录启用博客；不带参数时只包含本目录的文章
conf_enable_blog

# 聚合：把所有用户子目录下的 blog/ 视为同一个博客
conf_enable_blog users/*/blog
```

```
# 允许哪些组/用户发文，默认是 blog-editors 组
conf_blog_editors=(blog-editors)
```

```
# blog 首页上每篇文章保留前多少行（社区补丁，默认 7）
conf_max_lines_per_post=7
```

8.2 bridge ——给任何文档加评论

bridge 让任意文档/元素挂上评论或论坛式讨论：

```
# 在本目录及子目录开评论，允许 editors 组
conf_enable_comments editors

# -n 允许未注册用户评论，但需管理员审核
conf_enable_comments -n editors guests
```

8.3 dirdir ——一棵目录就是一个 wiki

dirdir 是 `diri` wiki 的下一代，搭建在 werc 之上：

- 层次化组织；
- Markdown 格式；
- 模板可定制；
- 用 werc 自己的用户/权限系统；
- 没有数据库；
- 启用只需在 `_werc/config` 写一个开关；
- 能把已有的 werc 文档树 “就地 wiki 化”；
- 实现仅约两打行 `rc` 脚本。

使用：登录之后到任意页面点 *Edit*。新建页面：访问目标地址，点 *Edit*，填内容，*Save*。（名字也是个文学梗：DirDir 既是 “diri 的迭代”，又是 Jack Vance 同名小说的致敬。）

8.3.1 Wiki 技巧：给 dirdir 加 “删除” 按钮

默认 dirdir 不提供删除按钮。社区给出的最小改动：

```
# sidebar_controls.tpl
<input type="submit" name="dirdir_edit"   value="Edit page" />
<input type="submit" name="dirdir_delete" value="Delete page" />
```

```
# apps/dirdir/app.rc :: dirdir_init
if(~ 1 $#post_arg_dirdir_edit $#post_arg_dirdir_preview)
    handler_body_main=(tpl_handler `{get_lib_file dirdir/edit.tpl apps/
       dirdir/edit.tpl})

# 新增删除分支
if not if(~ 1 $#post_arg_dirdir_delete)
    rm $dirdir_file

if not if(! ~ '' $"post_arg_dirdir_save $"post_arg_edit_text)
    save_page
```

修订目录被保留下来，相当于回收站。

8.4 duckduckgo ——用 DDG 做站内搜索

duckduckgo app 把一个站内路径（默认 `/_search/`）转成对 duckduckgo.com 的站内搜索查询。

启用流程：

```
# 1. 建立搜索目录及其 _werc/
mkdir -p /www/werc/sites/MYSITE/_search/_werc/

# 2. 启用 app
echo 'conf_enable_duckduckgo' \
    > /www/werc/sites/MYSITE/_search/_werc/config

# 3. 部署搜索框（示例：放进 footer）
mkdir -p /www/werc/sites/MYSITE/_werc/lib/
cp /www/werc/apps/search/footer.inc.sample \
   /www/werc/sites/MYSITE/_werc/lib/footer.inc
```

要看效果，去任何 cat-v.org 子站点的页脚瞧一眼搜索框即可。

8.5 wman ——把 man 页变成网站

wman 直接把现有的 Unix / Plan 9 man 页目录导出为网站，能识别 man 页之间的相互引用并自动生成链接，不需要预先把 man 转成 HTML。

```
# 把 /usr/share/man 暴露在当前路径下
conf_enable_wman /usr/share/man

# 若使用传统 Unix manN/foo.N 结构，请打开：
wman_unix_mode=1
```

演示：`http://man.cat-v.org`。

8.6 gregor ——卡夫卡式 Bug Tracker

gregor 是计划中的 bug 跟踪 app，名字来自 Kafka *Die Verwandlung* 的主人公 Gregor Samsa——某天早晨醒来发现自己变成了一只甲虫。目前仍处于设想阶段，官方页面列出的灵感来源包括 `bestpractical/sd` 与 viric 的 hgweb bug 仓库。

8.7 社区 apps

下面这些是 werc 用户写的、需要自取的外部 app（放进你 werc 的 `apps/` 即可）：

- **Xibit** ——soul9 写的图库 app；
- **SMAK** ——yiyus 的“非常简单”的图库；
- **hgwerc** ——yiyus 的 Mercurial Web 接口封装；
- **scrappydog** ——maht 的在线 scrapbook；
- **flip** ——yiyus 写的 PDF 查看 app；
- **barf** ——sl 写的博客 / 图床 / 日志 / paste 多合一站点，带分页与标签。

Chapter 9

案例与 Wiki 摘要

“In contrast to default werc setups we use 9base instead of plan9port.”

— suckless.org werc setup

9.1 suckless.org / sta.li 的部署

suckless 的 werc 部署是社区里讨论最多的一个“参考实现”。要点：

- 用 **9base** 替代 plan9port，把 Plan 9 工具栈压到最小；
- 用 **discount** 作为 Markdown 过滤器；
- 用 **nginx** + spawn-fcgi + fcgiwrap 来弥补 nginx 不支持原生 CGI 的限制；
- 修改 `lib/default_master.tpl`、`pub/style/style.css`、`etc/initrc.local`，实现横向 `sites_menu`；
- 移除 `top_bar.inc` 的包含——他们觉得跨站点条状导航是种视觉噪声。

具体片段已在第 6 章给出。

9.2 werchan：在自由软件 VPS 上跑社区站

werchan 是托管在 `http://guhnoo.org/` 的一个长期实验：

- 基础设施：$3/年的 MPServ Ubuntu 14.04 VPS，用 `trisquelize.sh` 在不重装的前提下迁移到 Trisquel 7.0 GNU/Linux；
- 服务栈：nginx + `cgd`（由 SysVinit 启动），CGI 走 cgd 而非 fcgiwrap；
- 在 werc 之上叠加了一连串“hack”：评论编号、自写的 Markdown 处理器、移动 UI、一次性 Markdown 处理、wiki 页面创建辅助、风格表切换…

它示范了一件事：werc 不是只能跑文档站，把 `rc` + 文件系统 + 一点点 hack 拼起来足以承担一个长期的小型社区。

9.3 在 DreamHost 上活下来

第 3 章给出的.htaccess 配方就来自这则 Wiki。值得记住的两点：

- 把 `plan9port` 装到家目录；
- 引入一层 `werc.cgi` 壳脚本，让 `.cgi`（共享主机普遍认得）来调用 `werc.rc`。

Chapter 10

开发：版本、路线图、与 TODO

"There is no such thing!"

— werc 的版本编号哲学

10.1 获取开发代码

最新开发代码：

```
hg clone https://code.9front.org/hg/werc/
# 镜像：
git clone git://shithub.us/sl/werc
```

bug 报告、特性请求、补丁请发到 werc 邮件列表（见 "联系" 一章）。

10.2 版本编号

werc 没有严肃的版本编号约定。最初仿照 Linux 内核的偶数 = 稳定/奇数 = 开发，但实际开发更倾向于在单一分支上做增量改动。"激进" 的、可能破坏向后兼容的改动会留在已被弃用且当前未维护的 `werc-dev` 仓里，直到时机成熟才会启用。

10.3 Werc 1.5.x（当前稳定）的目标

- *[已完成]* 用 HTML 5 flexbox 重写 CSS 布局；
- *[待整合]* 文件上传支持（maht 贡献）；
- *[待整合]* OpenID 支持（maht 贡献）；
- *[已完成]* 更完整的测试套件（`http://tst.cat-v.org`）；
- *[已完成]* 整合 9front 上活跃站点的改动；
- *[已完成]* 引入 `rc-httpd`；
- *[已完成]* 站点级 `headers.tpl` 的可选覆盖；

- *[待整合]* 加入 `apps/mdir`（khm 贡献）；
- *[待整合]* 加入 `apps/paste`（khm 贡献，需基于 apps/upload 重写）。

10.4 Werc 1.4.x / 1.2.x / 1.1.x / 1.0.x

- 1.4.x：以 bug 修复、文档、小改进为主。
- 1.2.x：只接受安全修复。
- 1.1.x（开发分支）：所有模板与页面转 HTML 5；yuyis 的纯 awk Markdown 实现；原生 Plan 9 httpd 开箱即用；Google PubSubHubbub 支持（实验实现已完成）；新增测试套件。
- 1.0.x：已废弃，请升级。

10.5 Werc 2.0 计划

- 大规模重设计 / 重写；
- 所有破坏向后兼容的改动；
- 给那套函数与配置变量的命名“重新建立秩序”（“naming insanity” 是他们的原话）；
- 暂无其它计划。

10.6 长期 TODO 摘要

下面摘录 werc 开发页公开列出的若干想法（不全是承诺）：

- URL 规范化与重定向：去掉多余的 `.`、`,`、尾点，让邮件粘贴的链接更友好（“RC1 中基本完成”）；
- 博客：评论树（线程化）；分页（“部分完成”）；
- Sitemap：用 index 页作为目录描述；缓存生成（“RC0 中完成”）；
- 布局：让侧栏排序可控；提供右侧栏；让禁用全部侧栏 / 页眉 / 页脚（全屏模式）更容易；移动浏览器进一步测试与优化；
- 更好的页面 `<title>`（包括完整路径层级，“部分完成”）；
- 路径中允许 UTF-8 字符（要保证安全）；
- 写完整的回归测试套件；
- 替换掉所有对非 p9p / 非 Plan 9 程序的依赖（“可能已完成”）；
- 更好地为子 app 定义“API”（哪些环境变量、哪些函数可依赖）；
- 生成更好的 `<meta>`（description / keywords）；
- 把 werc 加入 Wikipedia 的 CMS / wiki 比较条目；
- 把 txt2tags 作为内建格式选项；
- 反垃圾：数学伪 CAPTCHA、隐形 honeypot 输入框（“Don’t write in this input box”）等；

- 远期：标签（写入 `_werc/tags`）、相关链接侧栏、bug tracker（gregor）、hg/git 仓库浏览器、AtomPub、9P 接口、静态站点生成、用 C 或 awk 写的更简洁的新 Markdown 实现。

Chapter 11

实战清单：从 0 到 1

下面这份清单适合做第一次部署时的检查表：

1. 准备 **Plan 9** 工具链。选 plan9port、9base、frontport 中的一个，在 Linux 上做静态链接（见第 2 章 “静态链接 Plan 9 from User Space”）。
2. 解包 **werc**。决定它的根目录，例如 `/var/www/werc`。
3. 复制一个示例站点。从 `sites/default.cat-v.org/` 复制一份给你自己的域名。
4. 写第一个 **`index.md`**。`echo '# Hello' > index.md` 就够了。
5. 配置 **HTTP** 服务器。选 lighttpd / Apache 2 / nginx / rc-httpd 中的一种，把 `.rc` 注册为 CGI、404 转 `werc.rc`、关闭服务器自带目录列表。
6. 用 **`addwuser.rc`** 创建账号。加入 `admin` 与你计划用到的角色组。
7. 在站点 **`_werc/config`** 里启用 **app**。例如博客：
   `conf_enable_blog` + `conf_enable_comments editors`；
   或者把整棵目录 wiki 化：`conf_enable_dirdir`。
8. 贴一个站点 **CSS** 覆盖。在 `_werc/pub/style.css` 里调主色。
9. 把目录 **`rsync`** 进版本控制系统。你的站点现在等同于一棵可 `git` 的目录树。

FAQ

A.1 自定义 favicon

Q 怎么给某个域名设自定义 favicon?

A 把 favicon 放在该域名根下，名为 `/favicon.ico` 即可；或用 HTTP 服务器的 alias / rewrite 把这个路径指向你想要的文件。

A.2 加新的 handler

Q 我想加一个像 `md_handler`、`tpl_handler` 那样的 handler，应该改 `corehandlers.rc` 吗？还有别的办法吗?

A 看 handler 的用途与受众：

- 一次性 / 站点特定的 hack：写在 `etc/initrc.local` 或 `_werc/config` 中即可，既定义又启用；
- 可被多人复用、复杂度较高、有模板依赖的：做成 `/apps/<name>/`，按照惯例提供 `conf_enable_<name>`；
- 通用且不太复杂的：加进 `corehandlers.rc`，发补丁给上游。

A.3 需要会 `rc` 才能用 werc 吗?

Q 我必须懂 `rc` 才能用 werc 吗?

A 不必。要扩展或写新 app 才需要 `rc`；对于标准部署，会 `mkdir` 与一个文本编辑器就够。

参考链接

B.1 官方资源

- 项目主页：http://werc.cat-v.org
- 文档：http://werc.cat-v.org/docs/
- Apps：http://werc.cat-v.org/apps/
- Wiki：http://werc.cat-v.org/wiki/
- 开发：http://werc.cat-v.org/development/
- Roadmap：http://werc.cat-v.org/development/roadmap.html
- TODO：http://werc.cat-v.org/development/todo.html
- 仓库：https://code.9front.org/hg/werc/

B.2 Plan 9 与 rc

- rc.cat-v.org：http://rc.cat-v.org（Tom Duff 的 *rc* 论文）
- man.cat-v.org：http://man.cat-v.org/plan_9/1/rc
- Plan 9 from User Space：https://9fans.github.io/plan9port/
- 9base：https://tools.suckless.org/9base/
- frontbase：https://code.9front.org/hg/frontbase

B.3 联系

- 邮件列表：发 *subscribe* 到 werc-owner@cat-v.org，之后即可发文到 werc@cat-v.org；
- commit 列表：发 *subscribe* 到 werc-commits-owner@cat-v.org；
- IRC：irc.oftc.org 的 #cat-v 频道。

用户评价（节选）

> *“When people ask me what I look for in a woman, I’m going to say, ‘werc-like qualities’.”*
> — shardz

> *“Okay, werc gets my official Mark of Sweetness. This is really damn nice.”*
> — Athas

> *“Thanks to Uriel and co. for providing the wonderful werc framework.”*
> — sleepydog, maintainer of Xpilot-AI

“Powered by werc.”