

目录

前言	1.1
好用的软件和工具	1.2
将手动变自动	1.3
自己写代码和工具	1.4
附录	1.5
参考资料	1.5.1

如何提高工作效率

简介

关于提高工作效率这个话题，内涵很广，此处不展开讨论。

此处只是探讨和分享，如何用各种工具和方法去提高工作的效率，减少重复劳动。

尤其是将一些手工处理的事情变成工具或脚本去自动处理，从而达到提高效率的目的。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitook源码

- [crifan/improve_work_efficiency: 如何提高工作效率](#)

在线浏览

- [如何提高工作效率 book.crifan.com](#)
- [如何提高工作效率 crifan.github.io](#)

离线下载阅读

- [如何提高工作效率 PDF](#)
- [如何提高工作效率 ePUB](#)
- [如何提高工作效率 MOBI](#)

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件
修订时间： 2018-01-16 16:50:00

好用的软件和工具

充分利用好用的合适的软件、工具，可以大幅提高自己的工作效率。

比如：

用notepad++的正则实现特殊的批量替换

具体效果可参考：

[3.4. Notepad++的正则表达式替换和替换 -- 【crifan推荐】轻量级文本编辑器，Notepad最佳替代品：Notepad++](#)

印象笔记同步到wordpress插件

利用

[印象笔记 Evernote 同步插件 for WordPress](#)

实现印象笔记中的内容同步到wordpress中

-> 否则我之前自己的印象笔记中有2000+的帖子的内容，想要发布到自己的wordpress网站crifan.com中，就只能自己一点点手动拷贝粘贴到wordpress后台的编辑器中去发布了，而且还要手动处理slug、标题、发布时间、一个个上传图片等等事情，就累死了。

-> 有了这个同步的插件，可以极大地提高内容发布的效率。

具体使用过程详见：

[【已解决】发布印象笔记的帖子到wordpress – 在路上](#)

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件
修订时间： 2018-01-16 17:14:20

将手动变自动

手动变自动的工具有哪些

为了实现提高工作效率，除了利用一些工具类的软件之外，其他还有一类，可以叫做 **脚本** 和 **小工具**。

从技术角度来说，脚本和小工具可以分为：

- shell
- makefile
- 命令行
 - 比如 npm 的命令行
- 脚本语言
 - 比如 python
- 其他小工具
 - 比如 awk , sed 等

手动变自动的实际例子

docbook的makefile 实现编译生成自动化

在折腾docbook期间，会涉及到输入docbook相关的很多命令，去将xml源码格式，转换为其他如 html, pdf, chm等多种格式，要输入的命令很多，很复杂，也很容易有笔误输错，为了提高准确度，更为了提供效率，减少重复劳动，所以去利用 `Makefile`，去写了makefile文件，实现将手动输入命令变成自动执行命令

makefile实现gitbook自动编译和发布

和上面的docbook中用makefile类似，此处也是折腾 `Gitbook` 期间，也是要输入很多命令，去执行 gitbook的调试，转换输出不同格式，所用也去写了 `Makefile` 去实现将繁杂的手动命令的输入，转换为自动执行，极大地减少了要输入的命令，提高了效率。

对应的Makefile的内容贴出来，供参考：

```
#####
# Global defines
#####

# COLORS
GREEN := $(shell tput -Txterm setaf 2)
YELLOW := $(shell tput -Txterm setaf 3)
WHITE := $(shell tput -Txterm setaf 7)
RESET := $(shell tput -Txterm sgr0)
```

```

# new line and tab
define NEWLINE

endifef

define TAB

endifef

#####
# Output current makefile info
#####
Author crifan.com
Version 20171231
Function Auto use gitbook to generated files: website/pdf epub/mobi; upload to remote serv
er; commit to github io repo
RunHelp = Run 'make help' to see usage
$(info -----
$(info ${YELLOW}Author${RESET} : ${GREEN}${Author}${RESET})
$(info ${YELLOW}Version${RESET} : ${GREEN}${Version}${RESET})
$(info ${YELLOW}Function${RESET}: ${GREEN}${(Function)}$(NEWLINE)${(TAB)}${(TAB)}$(RunHelp)${RES
ET})
$(info -----)

# get current folder name
# support call makefile from anywhere, not only from current path of makefile located
# MAKEFILE_PATH := $(abspath $(lastword $(MAKEFILE_LIST)))
# CURRENT_DIR_WITH_SLASH := $(notdir $(patsubst %,%,$(MAKEFILE_DIR)))
MAKEFILE_LIST_LASTWORD = $(lastword $(MAKEFILE_LIST))
MAKEFILE_PATH := $(abspath $(MAKEFILE_LIST_LASTWORD))
MAKEFILE_DIR := $(dir $(MAKEFILE_PATH))
MAKEFILE_DIR_PATSUBST := $(patsubst %,%,$(MAKEFILE_DIR))
MAKEFILE_DIR_NOSLASH = $(MAKEFILE_DIR_PATSUBST)
CURRENT_DIR_WITH_SLASH = $(MAKEFILE_DIR)
CURRENT_DIR = $(MAKEFILE_DIR_NOSLASH)
CURRENT_DIR_NAME := $(notdir $(MAKEFILE_DIR_PATSUBST))

BOOK_NAME := $(CURRENT_DIR_NAME)

OUTPUT_FOLDER_NAME = output
OUTPUT_PATH = $(CURRENT_DIR)/$(OUTPUT_FOLDER_NAME)/$(BOOK_NAME)
DEBUG_PATH = $(CURRENT_DIR)/debug

WEBSITE_PATH = $(OUTPUT_PATH)/website
PDF_PATH = $(OUTPUT_PATH)/pdf
EPUB_PATH = $(OUTPUT_PATH)/epub
MOBI_PATH = $(OUTPUT_PATH)/mobi

PDF_NAME = $(BOOK_NAME).pdf
EPUB_NAME = $(BOOK_NAME).epub
MOBI_NAME = $(BOOK_NAME).mobi

```

```

# ZIP_NAME = $(BOOK_NAME).zip

WEBSITE_FULLNAME = $(WEBSITE_PATH)
PDF_FULLNAME = $(PDF_PATH)/$(PDF_NAME)
EPUB_FULLNAME = $(EPUB_PATH)/$(EPUB_NAME)
MOBI_FULLNAME = $(MOBI_PATH)/$(MOBI_NAME)

.DEFAULT_GOAL := deploy

.PHONY : debug_dir debug
.PHONY : help
.PHONY : create_folder_all create_folder_website create_folder_pdf create_folder_epub crea
te_folder_mobi
.PHONY : clean_all clean_website clean_pdf clean_epub clean_mobi
.PHONY : all website pdf epub mobi
.PHONY : upload commit deploy

## Print current directory related info
.debug_dir:
    @echo MAKEFILE_LIST $(MAKEFILE_LIST)
    @echo MAKEFILE_LIST $(value MAKEFILE_LIST)
    @echo MAKEFILE_LIST_LASTWORD=$(MAKEFILE_LIST_LASTWORD)
    @echo MAKEFILE_PATH $(MAKEFILE_PATH)
    @echo MAKEFILE_DIR $(MAKEFILE_DIR)
    @echo MAKEFILE_DIR_PATSUBST=$(MAKEFILE_DIR_PATSUBST)
    @echo CURRENT_DIR_WITH_SLASH $(CURRENT_DIR_WITH_SLASH)
    @echo CURRENT_DIR $(CURRENT_DIR)
    @echo CURRENT_DIR_NAME $(CURRENT_DIR_NAME)
    @echo BOOK_NAME=$(BOOK_NAME)
    @echo OUTPUT_PATH $(OUTPUT_PATH)
    @echo WEBSITE_PATH $(WEBSITE_PATH)
    @echo WEBSITE_FULLNAME $(WEBSITE_FULLNAME)
    @echo PDF_PATH $(PDF_PATH)
    @echo PDF_FULLNAME $(PDF_FULLNAME)

#####
# Create folder
#####

## Create folder for gitbook local debug
.create_folder_debug:
    mkdir -p $(DEBUG_PATH)

## Create folder for gitbook website
.create_folder_website:
    mkdir -p $(WEBSITE_PATH)

## Create folder for pdf
.create_folder_pdf:
    mkdir -p $(PDF_PATH)

## Create folder for epub

```

```

create_folder_epub:
    mkdir -p $(EPUB_PATH)

## Create folder for mobi
create_folder_mobi:
    mkdir -p $(MOBI_PATH)

## Create folder for all: website/pdf/epub/mobi
create_folder_all: create_folder_website create_folder_pdf create_folder_epub create_folder_mobi

#####
# Clean
#####

## Clean gitbook debug
clean_debug:
    -rm -rf $(DEBUG_PATH)

## Clean generated gitbook website whole folder
clean_website:
    -rm -rf $(WEBSITE_PATH)

## Clean generated PDF file and whole folder
clean_pdf:
    -rm -rf $(PDF_PATH)

## Clean generated ePub file and whole folder
clean_epub:
    -rm -rf $(EPUB_PATH)

## Clean generated Mobi file and whole folder
clean_mobi:
    -rm -rf $(MOBI_PATH)

## Clean all generated files
clean_all: clean_website clean_pdf clean_epub clean_mobi

#####
# Generate Files
#####

## Debug gitbook
debug: clean_debug create_folder_debug
    gitbook serve $(CURRENT_DIR) $(DEBUG_PATH)

## Generate gitbook website
website: clean_website create_folder_website
    @echo
    @echo Generate website for $(BOOK_NAME)
    gitbook build $(CURRENT_DIR) $(WEBSITE_FULLNAME)

## Generate PDF file

```

```

pdf: clean_pdf create_folder_pdf
@echo
@echo Generate PDF for $(BOOK_NAME)
gitbook pdf $(CURRENT_DIR) $(PDF_FULLNAME)

## Generate ePub file
epub: clean_epub create_folder_epub
@echo
@echo Generate ePub for $(BOOK_NAME)
gitbook epub $(CURRENT_DIR) $(EPUB_FULLNAME)

## Generate Mobi file
mobi: clean_mobi create_folder_mobi
@echo
@echo Generate Mobi for $(BOOK_NAME)
gitbook mobi $(CURRENT_DIR) $(MOBI_FULLNAME)

## Generate all files: website/pdf/epub/mobi
all: website pdf epub mobi
@echo
@echo Generate All for $(BOOK_NAME)

# ######
# # Compress
# #####

# ## Compress all generated files to single zip file
# zip:
#     zip -r $(ZIP_NAME) $(OUTPUT_PATH)

# ## Clean compressed file
# clean_zip:
#     -rm -rf $(ZIP_NAME)

#####
# Upload to server
#####
PASSWORD_FILE ../sshpass_password.txt
REMOTE_USER root
REMOTE_SERVER 45.79.205.194
REMOTE_BOOKS_PATH /home/wwwroot/book.crifan.com/books
# REMOTE_PATH=$(REMOTE_BOOKS_PATH)/$(BOOK_NAME)
REMOTE_PATH $(REMOTE_BOOKS_PATH)

## upload all generated website/pdf/epub/mobi files to remote server using rsync. create s
shpass_password.txt file to contain password before use this
upload: all
@echo
@echo Upload for $(BOOK_NAME)
sshpass -f $(PASSWORD_FILE) rsync -avzh --progress --stats --delete --force $(OUTPUT_P
ATH) $(REMOTE_USER)@$REMOTE_SERVER:$REMOTE_PATH

```

```
#####
# Commit to github
#####
m ?= "1. update book ${BOOK_NAME}"
GITHUB_IO_PATH /Users/crifan/dev/dev_root/github/github.io/crifan.github.io

## Commit generated files to github io
commit: all
@echo
@echo Commit for ${BOOK_NAME}
rsync -avzh --progress --stats --delete --force ${OUTPUT_PATH} ${GITHUB_IO_PATH}
cd ${GITHUB_IO_PATH} && \
pwd && \
ls -la && \
git status && \
git add ${BOOK_NAME}/* && \
git status && \
git commit -m ${m} && \
git status && \
git push && \
cd ${CURRENT_DIR} && \
pwd

#####
# Deploy generated files to remote server and github.io repo
#####

## Deploy = Upload and Commit for generated files
deploy: upload commit
@echo
@echo Deploy for ${BOOK_NAME}

#####
# Help
#####

TARGET_MAX_CHAR_NUM=25
## Show help
help:
@echo ''
@echo 'Usage:'
@echo '  ${YELLOW}make${RESET} ${GREEN}<target>${RESET}'
@echo ''
@echo 'Targets:'
@awk '/^[_a-zA-Z\-\_0-9]+:/ { \
    helpMessage = match(lastLine, /### (.*)/); \
    if (helpMessage) { \
        helpCommand = substr($$1, 0, index($$1, ":")-1); \
        helpMessage = substr(lastLine, RSTART + 3, RLENGTH); \
        printf " ${YELLOW}%-${TARGET_MAX_CHAR_NUM}s${RESET} ${GREEN}%s${RESET}\n", he \
lpCommand, helpMessage; \
    } \
} \
}
```

```
{ lastLine = $$0 }' $(MAKEFILE_LIST)
```

而关于如何从无到有写出来这个Makefile的中间折腾过程，相关内容为：

- 【已解决】 makefile中获取当前文件夹名
- 【已解决】 给makefile添加make help时输出帮助信息
- 【已解决】 makefile中如何在输出信息时添加空格或Tab或换行
- 【已解决】 Mac中写Makefile或脚步实现自动化执行命令
- 【已解决】 rsync同步时使用--password-file但是没用仍需要输入密码
- 【已解决】 makfile执行任意命令时输出一些信息
- 【已解决】 用rsync去同步本地文件到远程服务器
- 【已解决】 命令行中ftp操作上传到远程服务器crifan.com
- 【已解决】 用github的io去存放个人的静态页面
- 【已解决】 rsync执行完毕终端仍偶尔显示进度条

TODO:

加上上述Makefile的具体的含义和解释

某前端Preact的基于npm的项目：用npm发布html页面到服务器

除了npm中 package.json 中的一些命令：

```
"scripts": {
  "dev": "cross-env NODE_ENV=development webpack-dev-server --inline --hot --progress",
  "start": "serve build -s -c 1",
  "prestart": "npm run build",
  "clean": "rm -rf build/",
  "prebuild": "npm run clean && mkdirp build && ncp src/assets build/assets",
  "build": "cross-env NODE_ENV=production webpack -p --progress",
  "lint": "eslint src"
},
```

之外，也去写了 Makefile 去实现自动将打包build出来的文件上传到服务器（的html的对应路径）上：

```
#####
# Global defines
#####

# COLORS
GREEN := $(shell tput -Txterm setaf 2)
YELLOW := $(shell tput -Txterm setaf 3)
WHITE := $(shell tput -Txterm setaf 7)
RESET := $(shell tput -Txterm sgr0)

# new line and tab
define NEWLINE
```

```

endif

define TAB

endiff

#####
# Output current makefile info
#####
Author crifan
Version 20171222
Function Auto deploy build files to remote server
RunHelp = Run 'make help' to see usage
$(info -----)
$(info ${YELLOW}Author${RESET} : ${GREEN}${Author}${RESET})
$(info ${YELLOW}Version${RESET} : ${GREEN}${Version}${RESET})
$(info ${YELLOW}Function${RESET}: ${GREEN}${Function}${NEWLINE}$(TAB)$(TAB)$(RunHelp)${RESET})
$(info -----)

# get current folder name
# support call makefile from anywhere, not only from current path of makefile located
# MAKEFILE_PATH := $(abspath $(lastword $(MAKEFILE_LIST)))
# CURRENT_DIR := $(notdir $(patsubst %,%,$(MAKEFILE_DIR)))
MAKEFILE_LIST_LASTWORD = $(lastword $(MAKEFILE_LIST))
MAKEFILE_PATH := $(abspath $(MAKEFILE_LIST_LASTWORD))
MAKEFILE_DIR := $(dir $(MAKEFILE_PATH))
MAKEFILE_DIR_PATSUBST := $(patsubst %,%,$(MAKEFILE_DIR))
MAKEFILE_DIR_NOSLASH = $(MAKEFILE_DIR_PATSUBST)
CURRENT_DIR = $(MAKEFILE_DIR)
CURRENT_DIR_NOSLASH = $(MAKEFILE_DIR_NOSLASH)
CURRENT_DIR_NAME := $(notdir $(MAKEFILE_DIR_PATSUBST))

REMOTE_FOLDER_NAME = uapp
BUILD_FOLDER_NAME = build
BUILD_PATH = $(CURRENT_DIR_NOSLASH)/$(BUILD_FOLDER_NAME)
TMP_PATH = $(CURRENT_DIR_NOSLASH)/$(REMOTE_FOLDER_NAME)

.DEFAULT_GOAL := deploy

.PHONY : debug_dir
.PHONY : help
.PHONY : create_folder_tmp copy_build
.PHONY : clean_tmp
.PHONY : clean_copy_deploy deploy

## Print current directory related info
debug_dir:
    @echo MAKEFILE_LIST $(MAKEFILE_LIST)
    @echo MAKEFILE_LIST $(value MAKEFILE_LIST)
    @echo MAKEFILE_LIST_LASTWORD $(MAKEFILE_LIST_LASTWORD)
    @echo MAKEFILE_PATH $(MAKEFILE_PATH)

```

```

@echo MAKEFILE_DIR ${MAKEFILE_DIR}
@echo MAKEFILE_DIR_PATSUBST ${MAKEFILE_DIR_PATSUBST}
@echo CURRENT_DIR ${CURRENT_DIR}
@echo CURRENT_DIR_NOSLASH ${CURRENT_DIR_NOSLASH}
@echo CURRENT_DIR_NAME ${CURRENT_DIR_NAME}
@echo REMOTE_FOLDER_NAME ${REMOTE_FOLDER_NAME}
@echo BUILD_FOLDER_NAME ${BUILD_FOLDER_NAME}
@echo BUILD_PATH ${BUILD_PATH}
@echo TMP_PATH ${TMP_PATH}

#####
# Create folder
#####

## Create folder for tmp folder
create_folder_tmp:
    mkdir -p ${TMP_PATH}

#####
# Clean
#####

## Clean tmp folder
clean_tmp:
    -rm -rf ${TMP_PATH}

## Clean tmp folder again
clean_tmp_2:
    -rm -rf ${TMP_PATH}

#####
# Deploy to server
#####

PASSWORD_FILE sshpass_password.txt
REMOTE_USER root
REMOTE_SERVER 42.123.123.254
REMOTE_PATH /opt/cowfarm/farm_web

## copy generated build files to tmp folder
copy_build: create_folder_tmp
    cp -a ${BUILD_PATH}/* ${TMP_PATH}

## Clean and Copy and Deploy
clean_copy_deploy: clean_tmp copy_build
    sshpass -f ${PASSWORD_FILE} rsync -avzh --progress --stats --delete --force ${TMP_PATH}
) ${REMOTE_USER}@${REMOTE_SERVER}: ${REMOTE_PATH}

## Deploy build files to remote server using rsync. Note: create sshpass_password.txt file
## to contain password before use this
deploy: | clean_copy_deploy clean_tmp_2

#####

```

```
# Help
#####
TARGET_MAX_CHAR_NUM 25
## Show help
help:
echo ''
echo 'Usage:'
echo '${YELLOW}make${RESET} ${GREEN}<target>${RESET}'
echo ''
echo 'Targets:'
@awk '/^[_a-zA-Z-_\0-9]+:/ { \
    helpMessage = match(lastLine, /### (.*)/); \
    if (helpMessage) { \
        helpCommand = substr($$1, 0, index($$1, ":")-1); \
        helpMessage = substr(lastLine, RSTART + 3, RLENGTH); \
        printf "$${YELLOW}%-${TARGET_MAX_CHAR_NUM}s${RESET} ${GREEN}%s${RESET}\n", he \
lpCommand, helpMessage; \
    } \
} \
{ lastLine = $$0 }' $(MAKEFILE_LIST)
```

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件
修订时间: 2018-01-16 17:16:10

自己写代码和工具

除了前面提到的用软件和工具去实现自动化以提高效率外，

如果实在找不到合适的软件和工具，则只能自己去写代码，写软件，去实现自己的目的了。

比如自己之前遇到过的：

自己写的同步国内博客帖子到wordpress的脚本：[BlogsToWordpress](#)

详见：

[crifan/BlogsToWordpress](#): 将(新版)百度空间, 网易163, 新浪sina, QQ空间, 人人网, CSDN, 搜狐Sohu, 博客大巴Blogbus, 天涯博客, 点点轻博客等博客搬家到WordPress

已无效 用于下载Songtaste中的歌曲： [DownloadSongtasteMusic](#)

之前喜欢听songtaste.com中的歌曲

后来自己用C#写工具，从其中下载自己喜欢的歌曲：

[crifan/downloadsongtastemusic](#)

不过现在songtaste.com网站已关闭了，所以此工具也就失效了。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件

修订时间：2018-01-16 17:17:08

附录

下面列出相关参考资料。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件
修订时间: 2018-01-16 11:50:00

参考资料

- 3.4. Notepad++的正则表达式替换和替换 -- 【crifan推荐】轻量级文本编辑器, Notepad最佳替代品: Notepad++
- 印象笔记 Evernote 同步插件 for WordPress
- 【已解决】发布印象笔记的帖子到wordpress – 在路上
- 【已解决】makefile中获取当前文件夹名
- 【已解决】给makefile添加make help时输出帮助信息
- 【已解决】makefile中如何在输出信息时添加空格或Tab或换行
- 【已解决】Mac中写Makefile或脚步实现自动化执行命令
- 【已解决】rsync同步时使用--password-file但是没用仍需要输入密码
- 【已解决】makfile执行任意命令时输出一些信息
- 【已解决】用rsync去同步本地文件到远程服务器
- 【已解决】命令行中ftp操作上传到远程服务器crifan.com
- 【已解决】用github的io去存放个人的静态页面
- 【已解决】rsync执行完毕终端仍偶尔显示进度条
- crifan/BlogsToWordpress: 将(新版)百度空间,网易163,新浪sina,QQ空间,人人网,CSDN,搜狐Sohu,博客大巴Blogbus,天涯博客,点点轻博客等博客搬家到WordPress
- crifan/downloadsongtastemusic

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件
修订时间: 2018-01-16 17:12:31