# Contents

# News

Mon May 16 01:41:35 PDT 2022 - move `vimdoc` to LEGACY (bottom) - add keycode, xmodmap

# TODO:

- See Index Cards (bottom) - integrate that point to very bottom into main text
  - HTTR2 notes - mess

PURPOSE: Misc Tech Notes; details, notes can be here - but COMMANDS put on INDEX

USING: pandoc, latex,

as of June 3, 2022:
* To mix latex and .md, must go with pdf, either pandoc or knit
* Add r, knitr code to YAML? then must render as .RMD file * I do not know how to embed latex, produce html or md (github flavor).
# PDF

PDF [ignores html, css; also ignores YAML header (pandoc & ::render()]

To create pdf, just about everything works: pandoc, markdown, latex, knitr..

NOTE: Missing latex .sty ?
With .tex file, run (in R) tinytex:latemk(*.tex) to install

fonts installed? fc-list : family (Oct 2021) Can not figure out how to use another font in pandoc: mainfont: is not working.

(Jan 2022) **Missing font, package? TinyTex** * update R * keep_tex: true (in YAML) * at R console tinytex::lualatex(“…. .tex”), or tinytex::latexmk(“*.tex”)*
~~ some times works, sometimes not ~~

Lua in *.tex file* see ~/code/publish_project/TEX/

~~~
```vimdoc
!pandoc % -f markdown -o %.pdf

!pandoc % -f markdown  -t latex -H ../chapter_break.tex -V linkcolor:blue -V fontsize=11pt -V geometry:margin=0.3in -o ~/Downloads/print_and_delete/out.pdf
!pandoc % -f markdown  --pdf-engine lualatex -H chapter_break.tex -V linkcolor:blue -V fontsize=11pt -V geometry:margin=0.3in -o ~/Downloads/print_and_delete/out.pdf

!pandoc --metadata=project:JIM --lua-filter doc/panvimdoc/scripts/skip-blocks.lua --lua-filter doc/panvimdoc/scripts/include-files.lua  -t doc/panvimdoc/scripts/panvimdoc.lua % -o doc/source/jim_knitr_pandoc_latex.txt
~~~

# HTML

```
  HTML [to produce HTML with pandoc, all latex is IGNORED.]

I do **not** know how to create fancy HTML files from knitr, pandoc.

 HTML is pain in ass and HUGE time waste.  Pandoc can handle markdown and
 small amounts of latex (math) b/c ppl have added filters or other widgets to
 pandoc.

 Avoid experiments:   will waste time.
 If using Latex, its packages, diagrams with Latex ... must go with PDF.


  *  !pandoc % -f markdown -V linkcolor:blue -V fontsize=11pt -V geometry:margin=0.3in -o out/out.html


-H header
-V or --variable
--pdf-engine=xelatex


*Create pdf from straight txt*
(do not process any markdown)

pandoc balks at processing straight text if it things it sees markdown.
If lucky, !pandoc % -o file.pdf will work.

**BEST**  print_me.sh *.txt file;  then use browser to print and save as .pdf
*.R  - NOPE, Firefox chokes.
```

# R, DEFINTIONS, TERSE EXAMPLES

```
Not in this document.
See ~/code/try_things_here/BASE/
```

# GIST, GITHUB download to R

```
download.file("https://gist.githubusercontent.com/jimrothstein/c5e148c9a766ab1a1a91464517a0fe1f/raw/61655207c75c898be9a80454abf53bee0
              destfile="junk.txt")
readLines(con="junk.txt")
```

# LATEX NOTES

- Tikz seems to be most popular way to gaphics.
- footnote: **postscript** is more powerful programming language; pdf hails from this. However, using postscript with latex requires addins, such as ghostscript; drivers; … Avoid postscript and packages pstricks, even if greater capability.

## neovim, nvim, vim update to latest version {

```
##  Tue  02 Nov 2021 (also 30 DEC 2021)
-  download nvim.appimage | place in ~/bin/ | will overwrite prior
-  change permissions to  764
-  do not touch soft link nvim --> nvim.appimage
-  nothing more than this.

##  Wed  09 Feb 2022

-   neovim TERMINAL BUFFER has 2 modes:  Normal (move around as usual, gf, y
etc) and a NEW MODE:  Terminal mode.  This mode means we see BASH cursor.
Anything entered goes there.  There is NO INSERT/EDIT Mode.  You deal with
Terminal mode at the ACTIVE line only.
See #75 Vimcast

-   This mapping copies line , inserts into terminal buffer and runs
noremap <leader>tl Vy<C-w>wpa<CR><C-\><C-n><C-w>pj


}
```

## VIM writing_notes

```
*jim_writing_notes1*

http://www.terminally-incoherent.com/blog/2013/06/17/using-vim-for-writing-prose/
:h help-writing
## hard wrap is friend

a=automatic reformat
t=wrap at textwidth

setlocal formatoptions=ant
setlocal textwidth=80
setlocal wrapmargin=0
setlocal foldcolumn=3       "trick, to set left margin


Long parapgarapja l;akdsjf asalkfjas d; asdfk;ladsjf  lk;adjf a;lkaf as;l
asdfjl; adsfl;kj d;as fasdj;lkj afds;lkj

##  Folds
26FEB2022  set to use treesitter;  don't seem to work

##  Turn off indents

(no c indents)

setlocal noautoindent
setlocal nocindent
setlocal nosmartindent
setlocal indentexpr=
```

## GIT commands

- Change git push from https to ssh

```
git remote -v   shows using https:
git remote set-url origin git@github.com:jimrothstein/REPOSITORY.git
git remote -v # shows using git:
```

## LUA

```
In lua, nil or false evaluate to:  false
0 or '', evaluate to: true


~/code/lua_project/
```

```
Lua + neovim:
    *  code is lua.
    *  but calls the neovim API | look careful, can see the vim
        *  api.nvim...command("enew") -- creates new file and edits.
        *  vim.bo[0],buftype=nofile

--  These are vim api ,  called by lua

-- [[  multi-
-- line
-- comments
-- ]]

-- shortcuts:
local cmd = vim.cmd
cmd("pwd")   -- execute vim Ex: command

-- current file name:
:lua print(vim.fn.expand('%')

--  set vim options
:lua vim.api.nvim_command('set nonumber')
:lua vim.api.nvim_command('set number!')        -- toggle
:lua vim.api.nvim_command('echo "Hello, Nvim!"')

--  list buffers, vim.cmd is alias for vim.api.nvim_exec()
:lua vim.cmd('buffers')

-- print
:lua print(_VERSION)
:lua print("hi")

-- print, datatypes
-- Data types are converted correctly
print(vim.api.nvim_eval('1 + 1')) -- 2
print(vim.inspect(vim.api.nvim_eval('[1, 2, 3]'))) -- { 1, 2, 3 }
print(vim.inspect(vim.api.nvim_eval('{"foo": "bar", "baz": "qux"}'))) -- { baz = "qux", foo = "bar" }
print(vim.api.nvim_eval('v:true')) -- true
print(vim.api.nvim_eval('v:null')) -- nil


vim.api.nvim_command('new')

-- To run a lua file
:luafile %
x = 41
if x > 40 then
  print('over 40')
else
  print('under')
end

-- verb (in init.vim)
-- y{motion} will highlight for you!
-- :au TextYankPost * silent! lua vim.highlight.on_yank()

-- This is a .lua file
-- To source it from .vim:   :luafile <file>
-- :luafile % will also work.
x = "hello"
print(x)

-- tools.lua
local api = vim.api
local M = {}
function M.makeScratch()
  api.nvim.command('enew') -- equal to :enew
  vim.bo[0].buftype=md
end
return M


-- in vim
-- create new command (fails)
-- :command! Scratch lua require'0001_tools'.makeScratch()
--
```

```
-- :lua vim.wo.number = true
-- vim.api.nvim_set_win_option('number', true)
-- lua print(vim.wo.number)

-- in a lua file only need following  (and reload)
vim.wo.number = true
vim.wo.number = false
vim.bo.shiftwidth = 4
```

# VIMDOC

## SOURCE md file

The SOURCE markdown file is located in jimHelp/source. Edit .md file; not the resulting .txt file. Edit .md file; not the resulting .txt file

## Resulting txt file.

The resulting txt file will be located in jimHelp/doc

# Vim Notes

HELPTAGS and Ctags are NOT related (do not confuse).

for ctags: :h tags-file-format

To change file: edit this file as regular file. Dislike Highighting? :set syntax=off Add a tag: surround new tag with * ; plus prose to describe tag Add a hotlink: ONLY in same file (I think) surround new tag with |

Run :helptags ALL to regenerate file called tags /doc file (singular) : should see this .txt file and tags file

Following sets things up: * open .R file * start R (should be bottom) * :vert h (open help on right)

## VIM help 1

```
:h windows.txt
:h vert
:h splitright


:h new   " open new WINDOW
:h enew  " new buffer, in current window

*jim_system_stuff*
:view $VIMRUNTIME
:view $TEMPLATES

*jim_auto_commands*
:h autocmd
:h au



[all docs files](~/docs/)
[code files](~/code/)

:h abbreviation
:h help-summary
:h helphelp
:h help.txt
:h helpgrep
:h usr_toc.txt
:h index
```

```
:h startup
:h cmdline
:h exe    (use cmd line to run normal cmds?)
:h startinsert


####  Help for common tasks

:h :abbreviate
:h :augroup
:h :changes
:h :highlight
:h :syntax
:h :command
:h :file
:h :filetype
:h :messages
:h :options  :h options.txt  :h :set
:h :omni
:h :complete  "NOTE:  nvim does NOT have cmd-line completion like C-N, C-P
:h map-listing

:h :scriptnames
:h man  (use vim for manpages)


*jim_split*
:h :split
:vert help    " open help in vertical split

(N) !!date, insert date

:resize -3 <CR>  " reduce size of window
:vertical resize -3 <CR>

$VIMRUNTIME (inside the image app)
:!ls $VIMRUNTIME

## Windows, splits
:h usr_07.txt
:h usr_08.txt
:h windows.txt
:h CTRL-W


## statusline  %m (modify?) %y (filetype) ...
:h statusline
:echo expand("%m")
:set statusline=%t
:set statusline+=%{&ff}

Ranges (in file)
:h range
:., 'a
:., +2
3 lines below to end - 5 lines
:.+3, $-5

## insert mode
:h insert.txt
:h insert-index
:h i_CTRL-R

<C-R>% inserts file name:
/home/jim/docs/misc_files/005_tech_notes.md

<C-R>=system("ls")  inserts listing


Insert in bulk:
:i or :a  followed by . when done


## Registers
:echo @a
:let @a="hello"
```

```
## Plugins
:h Vimux
:call VimuxRunCommand("ls")
:VimuxPromptCommand<CR>

To Close:
:VimuxCloseRunner<CR>


## Syntax Highlighting
:h usr_06.txt

## vim initialize
:vert h nvim_R
:tab help


## vim help 2
:vert h nvim-R  " opens help to right
:let R_nvimpager = "vertical" default, (can be "tab", "tabnew")

## vim & grep (search both *.R and *.Rmd - note | is escaped)
:grep -EHRn 'binomial' ~/code/**/*.(R\|Rmd)

## vim tabs
tabs   :tabn :tabp :tabnew

READ: cmds to open windows at various localations:  bo, above ...


:h reference_toc
:h help
:h help-summary
:h cmd   (:h ls)
:helpgrep fold*  (no quotes)

"all tags
:h quickref.txt

"index
:h usr_toc.txt

:h reference_toc   (all *.txt files)
:h local-additions (plugins)

:h motions.txt (jumps, motions, find next } etc)


## search
    /foo/+1    find foo  and move +1 line down
/foo/0     find .... but move to beginning of line
/foo/e-1    find ... then move back 1 character.
```

## VIM HELP 3 (context)

```
:h i_{}      (insert, delete, visual, ...)

:h :ex_cmd

:h 'option'

:h func()

:h /[     (escape regex character)

:h ft-r-indent    (for plugin r)
:h ft-json-....   (for plugin json)
```

# Pandoc Notes

```
as of \today:
    *  To mix latex and .md, must go with pdf, either pandoc or knit
    *  Add r, knitr code to YAML?   then must render as .RMD file
    *  I do not know how to embed latex, produce html or md (github flavor).

 ===
  PDF   [ignores html, css; also ignores YAML header (pandoc & ::render()]
 ===

  *  Pandoc creates a .tex file (from .md source).   This .tex file is run
      through engine (pdflatex, xelatex ....) to actually output the pdf.

  NOTE:   .tex uses a .sty which I do not have.  USE  knitr:: (with TinyTex
  to locate and install that .sty file)


!pandoc % -f markdown -o %.pdf

!pandoc % -f markdown  -t latex -H ../chapter_break.tex -V linkcolor:blue -V fontsize=11pt -V geometry:margin=0.3in -
o ~/Downloads/print_and_delete/out.pdf
!pandoc % -f markdown  -t latex -H ../chapter_break.tex -V linkcolor:blue -V fontsize=11pt -V geometry:margin=0.3in -
o out.pdf
!pandoc % -f markdown  --pdf-engine lualatex -H chapter_break.tex -V linkcolor:blue -V fontsize=11pt -V geometry:margin=0.3in -
o ~/Downloads/print_and_delete/out.pdf

 ====
  HTML [ignores latex]
 ====

  *  !pandoc % -f markdown -V linkcolor:blue -V fontsize=11pt -V geometry:margin=0.3in -o out/out.html
```

### LINUX/ZSH notes

Wed May 25 20:22:20 PDT 2022 - run GallumOS 18.04 – `password for Ubuntu, UbuntuOne is icmup.6667.again` - Ubuntu could NOT install (wifi issues) - But old laptop runs fine with just linux mode.

### sudo vs su …. `{`

```
-   su jim  change to User 'jim'
- sudo cmd
    -   last ~ 15'  (temporary use of root privileges)
    -   asks for user's password
    -   allows root 'privileges' but the home directory, path etc remains the
        user's
- sudo su   # run cmd su (to switch user) with root permissions. (default is
    root)
- **sudo su -** # run cmd su (to change user) with root permissions AND WITH root
    environment (echo $SHELL will root)
-   shell:  either login or non-login
-   non-login has 2 flavors: **interactive** (user at CLI) and **non-interactive** (a
    subshell for scripts)
```

!askubuntu 376199 !askubuntu 1225041

} {} #### drive info { # succinct, useful info lsblk –output NAME,UUID,PARTUUID }

### xev

{ - Keyboard specific, find what *keycode* a button is mapped to: - USAGE: > xev - type just 1 button, look for its keycode, keysym on this keyboard - example: q will be keycode=24, keysym=0x71 called ‘q’

} #### xxd { - To find how zsh maps a button (A, alt, F2) :
- USAGE: > xxd - press +a - terminal displays coding (^[a) - SEE ROTHGAR } #### remap capslock to escape

{ # PURPOSE: **maps ChromeBox “capslock” key to Escape.** # - use > xev to find that capslock is key 133. # - xmodmap is older, but simpler to change key action to change key action. # - newer is **setxkbmap** but I find more effort to figure out simple things. # - SEE tech_notes # - lots of ways to do this remap. This works, stay with it: # xmodmap -e “keycode 133 = Escape” }

{ grep jim /var/log/syslog # see cron jobs that ran

Sat May 21 18:48:16 PDT 2022 - jr changed /etc/rsyslog/50-default.conf - uncomment #cron – cron s/d now log to cron.log
- after change, run sudo service rsyslog restart

}

**Linux Kernel** { - one LTS Ubuntu can have many (upstream) kernels - Mix & Match kernels? X? - Kernel Upgrade - See INDEX C.

}

```
*jim_Permissions*
u g o   (user group other)



*grep_vs_ls*
*Grep* always finds words that match a pattern and returns file names of
matches.

ls (+ glob) finds filenames that match a pattern.  Very different.
(same in vim)

*jim_GLOB_examples*
Mostly of form ls or ll or print -l    and **/*
example:   print -l ~/code/**/*.(R|Rmd)   # any level, return all .R and .Rmd
files

See my zsh GLOG handwritten notes (till typed in here)

*zle_widgets* (all commands)
Output from zle -al (~403 cmds)
.accept-and-hold
.accept-and-infer-next-history
.accept-and-menu-complete
.accept-line
.accept-line-and-down-history
.accept-search
.argument-base
.auto-suffix-remove
.auto-suffix-retain
.backward-char
.backward-delete-char
.backward-delete-word
.backward-kill-line
.backward-kill-word
.backward-word
.beep
.beginning-of-buffer-or-history
.beginning-of-history
.beginning-of-line
.beginning-of-line-hist
.bracketed-paste
.capitalize-word
.clear-screen
.complete-word
.copy-prev-shell-word
.copy-prev-word
.copy-region-as-kill
.deactivate-region
.delete-char
.delete-char-or-list
.delete-word
.describe-key-briefly
.digit-argument
.down-case-word
.down-history
.down-line
.down-line-or-history
.down-line-or-search
.emacs-backward-word
.emacs-forward-word
.end-of-buffer-or-history
.end-of-history
.end-of-line
.end-of-line-hist
.end-of-list
.exchange-point-and-mark
.execute-last-named-cmd
.execute-named-cmd
.expand-cmd-path
.expand-history
.expand-or-complete
```

```
.expand-or-complete-prefix
.expand-word
.forward-char
.forward-word
.get-line
.gosmacs-transpose-chars
.history-beginning-search-backward
.history-beginning-search-forward
.history-incremental-pattern-search-backward
.history-incremental-pattern-search-forward
.history-incremental-search-backward
.history-incremental-search-forward
.history-search-backward
.history-search-forward
.infer-next-history
.insert-last-word
.kill-buffer
.kill-line
.kill-region
.kill-whole-line
.kill-word
.list-choices
.list-expand
.magic-space
.menu-complete
.menu-expand-or-complete
.neg-argument
.overwrite-mode
.pound-insert
.push-input
.push-line
.push-line-or-edit
.put-replace-selection
.quote-line
.quote-region
.quoted-insert
.read-command
.recursive-edit
.redisplay
.redo
.reset-prompt
.reverse-menu-complete
.run-help
.select-a-blank-word
.select-a-shell-word
.select-a-word
.select-in-blank-word
.select-in-shell-word
.select-in-word
.self-insert
.self-insert-unmeta
.send-break
.set-local-history
.set-mark-command
.spell-word
.split-undo
.transpose-chars
.transpose-words
.undefined-key
.undo
.universal-argument
.up-case-word
.up-history
.up-line
.up-line-or-history
.up-line-or-search
.vi-add-eol
.vi-add-next
.vi-backward-blank-word
.vi-backward-blank-word-end
.vi-backward-char
.vi-backward-delete-char
.vi-backward-kill-word
.vi-backward-word
.vi-backward-word-end
.vi-beginning-of-line
```

```
.vi-caps-lock-panic
.vi-change
.vi-change-eol
.vi-change-whole-line
.vi-cmd-mode
.vi-delete
.vi-delete-char
.vi-digit-or-beginning-of-line
.vi-down-case
.vi-down-line-or-history
.vi-end-of-line
.vi-fetch-history
.vi-find-next-char
.vi-find-next-char-skip
.vi-find-prev-char
.vi-find-prev-char-skip
.vi-first-non-blank
.vi-forward-blank-word
.vi-forward-blank-word-end
.vi-forward-char
.vi-forward-word
.vi-forward-word-end
.vi-goto-column
.vi-goto-mark
.vi-goto-mark-line
.vi-history-search-backward
.vi-history-search-forward
.vi-indent
.vi-insert
.vi-insert-bol
.vi-join
.vi-kill-eol
.vi-kill-line
.vi-match-bracket
.vi-open-line-above
.vi-open-line-below
.vi-oper-swap-case
.vi-pound-insert
.vi-put-after
.vi-put-before
.vi-quoted-insert
.vi-repeat-change
.vi-repeat-find
.vi-repeat-search
.vi-replace
.vi-replace-chars
.vi-rev-repeat-find
.vi-rev-repeat-search
.vi-set-buffer
.vi-set-mark
.vi-substitute
.vi-swap-case
.vi-undo-change
.vi-unindent
.vi-up-case
.vi-up-line-or-history
.vi-yank
.vi-yank-eol
.vi-yank-whole-line
.visual-line-mode
.visual-mode
.what-cursor-position
.where-is
.which-command
.yank
.yank-pop
_bash_complete-word
_bash_list-choices
_complete_debug
_complete_help
_complete_tag
_correct_filename
_correct_word
_expand_alias
_expand_word
_history-complete-newer
```

```
_history-complete-older
_list_expansions
_most_recent_file
_next_tags
_read_comp
accept-and-hold
accept-and-infer-next-history
accept-and-menu-complete
accept-line
accept-line-and-down-history
accept-search
argument-base
auto-suffix-remove
auto-suffix-retain
backward-char
backward-delete-char
backward-delete-word
backward-kill-line
backward-kill-word
backward-word
beep
beginning-of-buffer-or-history
beginning-of-history
beginning-of-line
beginning-of-line-hist
bracketed-paste
capitalize-word
clear-screen
complete-word
copy-prev-shell-word
copy-prev-word
copy-region-as-kill
deactivate-region
delete-char
delete-char-or-list
delete-word
describe-key-briefly
digit-argument
down-case-word
down-history
down-line
down-line-or-history
down-line-or-search
emacs-backward-word
emacs-forward-word
end-of-buffer-or-history
end-of-history
end-of-line
end-of-line-hist
end-of-list
exchange-point-and-mark
execute-last-named-cmd
execute-named-cmd
expand-cmd-path
expand-history
expand-or-complete
expand-or-complete-prefix
expand-word
forward-char
forward-word
get-line
gosmacs-transpose-chars
history-beginning-search-backward
history-beginning-search-forward
history-incremental-pattern-search-backward
history-incremental-pattern-search-forward
history-incremental-search-backward
history-incremental-search-forward
history-search-backward
history-search-forward
infer-next-history
insert-last-word
kill-buffer
kill-line
kill-region
kill-whole-line
```

```
kill-word
list-choices
list-expand
magic-space
menu-complete
menu-expand-or-complete
neg-argument
overwrite-mode
pound-insert
push-input
push-line
push-line-or-edit
put-replace-selection
quote-line
quote-region
quoted-insert
read-command
recursive-edit
redisplay
redo
reset-prompt
reverse-menu-complete
run-help
select-a-blank-word
select-a-shell-word
select-a-word
select-in-blank-word
select-in-shell-word
select-in-word
self-insert
self-insert-unmeta
send-break
set-local-history
set-mark-command
spell-word
split-undo
transpose-chars
transpose-words
undefined-key
undo
universal-argument
up-case-word
up-history
up-line
up-line-or-history
up-line-or-search
vi-add-eol
vi-add-next
vi-backward-blank-word
vi-backward-blank-word-end
vi-backward-char
vi-backward-delete-char
vi-backward-kill-word
vi-backward-word
vi-backward-word-end
vi-beginning-of-line
vi-caps-lock-panic
vi-change
vi-change-eol
vi-change-whole-line
vi-cmd-mode
vi-delete
vi-delete-char
vi-digit-or-beginning-of-line
vi-down-case
vi-down-line-or-history
vi-end-of-line
vi-fetch-history
vi-find-next-char
vi-find-next-char-skip
vi-find-prev-char
vi-find-prev-char-skip
vi-first-non-blank
vi-forward-blank-word
vi-forward-blank-word-end
vi-forward-char
```

```
vi-forward-word
vi-forward-word-end
vi-goto-column
vi-goto-mark
vi-goto-mark-line
vi-history-search-backward
vi-history-search-forward
vi-indent
vi-insert
vi-insert-bol
vi-join
vi-kill-eol
vi-kill-line
vi-match-bracket
vi-open-line-above
vi-open-line-below
vi-oper-swap-case
vi-pound-insert
vi-put-after
vi-put-before
vi-quoted-insert
vi-repeat-change
vi-repeat-find
vi-repeat-search
vi-replace
vi-replace-chars
vi-rev-repeat-find
vi-rev-repeat-search
vi-set-buffer
vi-set-mark
vi-substitute
vi-swap-case
vi-undo-change
vi-unindent
vi-up-case
vi-up-line-or-history
vi-yank
vi-yank-eol
vi-yank-whole-line
visual-line-mode
visual-mode
what-cursor-position
where-is
which-command
yank
yank-pop
zle-line-finish
zle-line-init
```

## BINDKEY

```
*bindkey*  # results, all shortcuts
```

```
"^A"-"^C" self-insert
"^D" list-choices
"^E"-"^F" self-insert
"^G" list-expand
"^H" vi-backward-delete-char
"^I" expand-or-complete
"^J" accept-line
"^K" self-insert
"^L" clear-screen
"^M" accept-line
"^N"-"^P" self-insert
"^Q" vi-quoted-insert
"^R" redisplay
"^S"-"^T" self-insert
"^U" vi-kill-line
"^V" vi-quoted-insert
"^W" vi-backward-kill-word
"^X^R" _read_comp
"^X?" _complete_debug
"^XC" _correct_filename
"^Xa" _expand_alias
```

```
"^Xc" _correct_word
"^Xd" _list_expansions
"^Xe" _expand_word
"^Xh" _complete_help
"^Xm" _most_recent_file
"^Xn" _next_tags
"^Xt" _complete_tag
"^X~" _bash_list-choices
"^Y" self-insert
"^Z" backward-delete-word
"^[" vi-cmd-mode
"^[," _history-complete-newer
"^[/" _history-complete-older
"^[OA" up-line-or-history
"^[OB" down-line-or-history
"^[OC" vi-forward-char
"^[OD" vi-backward-char
"^[[1~" vi-beginning-of-line
"^[[200~" bracketed-paste
"^[[2~" overwrite-mode
"^[[3~" vi-delete-char
"^[[4~" vi-end-of-line
"^[[A" up-line-or-history
"^[[B" down-line-or-history
"^[[C" vi-forward-char
"^[[D" vi-backward-char
"^[~" _bash_complete-word
"^\\\\"-"~" self-insert
"^?" vi-backward-delete-char
"\M-^@"-"\M-^?" self-insert
```

# XFCE4

Shortcuts: https://docs.xfce.org/apps/terminal/start#keyboard_shortcuts HELP: https://docs.xfce.org/apps/terminal/4.12/start

Based on VTE Widget terminal (gnome uses)

ALT-F10 toggle bet min/max (NOPE!)

ALT-TAB rotate through open windows?

```
##  Thu  19 Nov 2020  Acer Batttery
  *  ACER CB3-431-C7EX
  *  From back (tiny print on labels)
    *  SNID   8120 1450072
    *  SN NXGC7AA001812038A47200
    *  ACER CB-431 Model N16P1
Do you sell new battery for this ACER laptop?
CB3-431-C7EX  (manuf 3/22/18)
SNID:  81201450072
```

### REST RESTful, HTTP Protocol

```
    -   HTTP - best is Mozilla introduction
  - In practice, Restful API means built upon HTTP. (do exist non-HTTP )
*   originally URL linked to file or webpage.
*   more recenty, URI links to payload,  HTML/JSON/XML
*   RESTFUL provides stateless operations, architecture (vs SOAP, or others)
* VERBS include GET/POST/ etc etc
Stateless means server keeps no session information.   Each call to server is
independent.  Examples include HTTP, IP, REST.   But TCP is not stateless.

### epub, Calibre, iPad, iCloud, eReader, pdf
-   Claim:  iPad does not support Calibre; free Readers for iPad, everyone has fav.
        No, no, no.    Download Calibre software for osx to iPad.   What  does not
        work is connecting iPad to Calibre on Laptop.
-   Goodreader for pdf ($20?) - many say best iPad reader.?
- Marvin - no pdf support, but excellent otherwise?
```

# KNITR

```
**knitr -> R & rmarkdown -> Bookdown (~2016) -> Blogdown -> netlify (Hugo, static)**
HUGO:   md -> html
BOOKDOWN:  Rmd        ->html (skips md)

**lua** is a lightweight language acts like "glue" ; embeds within code; useful in
textdoc .

**renv**    Why I think I do not need (and do not want).  Re-creates tidyverse code
INSIDE each project, ie local copy of everything inside package.  Then takes
snapshots as either your code or the any of like libraries changes.   Nice
purpose:   easily re-create complete environment.   But much too much overhead
for my needs!  (at this time.)
```

# X11

```
-   XFCE - many distros, suite of apps, use GTK+ toolkit
-   -   DESKTOP Mgr=Xfdesktop (colors, images, wallpaper)
-   -   FILE Mgr=Thunar (GTK+ toolkit)
        -   others: nautilus
-   -   Windows mgr=xfwm4 (max, min, focus, tiling ...)
-   -   Settings mgr=xfce4-settings-manager (appearance, style, keyboard, ....)
-   -   Terminal=xfce4-terminal (1 of many possible emulators, code that sits
          inside bash?)
-   DISPLAY MGR (DM) = Begins X, then displays (gui) login screen.  Many types
    of DM.
-   chroot  -   Without rebooting, chroot means "change root" ie start new
    shell, change root diretory (to point to a partition)

-   X uses(?) xlib (old), xcb(newer)
-   ncurses lib -?
-   Wayland - next generation (replace?) for X
-   Stack - X at bottom, GNOME or KDE above, NAUTILUS or panels above
-   man Xorg (good) , I have no ~/.xinitrc
-   Terminal is NOT equal to SHELL (explain?)
-   GTK+ - C lib, widgets supports X.   Gnone, Win32, etc use GTK+ tools.
-   [see wiki] GTK is C toolkit, widgets (now gtk3, soon gtk4)
- I have GTK, competition is qt
-   graphical login? kdm, gdm, xdm (basic) lightdm, sddm aka Display Mgr
-   REMOVE PLUGIN:  vimwiki - how to get rid  | .vimrc - delete references to plugin
##  13 OCT 2018
-   Working: Ranger, newsbeut , updated to Ubuntu 18.04LTS
-   TERMINALS
    -   rxvt, urxvt, terminator, st (not friendly) xfce4-terminal.
```

# CURL

Purpose: Examples of Curl at CLI, references to more details.

Note: Using curl with Youtube API is separate (see below)

## CURL + youtube api

Sat 02 Apr 2022

PURPOSE: Focus is Google API, youtube in particular. This is summary of using cURL to obtain authorization_code and then proceed querying youtube.

USAGE: This is a markdown, md, file. All zsh code is treated as verbatim. To run the zsh, use the neovim terminal and with short cut ,tl.

Once something is working convert to a zsh script file. But THIS document be NEAT summary.

- ~/.Renviron for secrets
  * Source: https://developers.google.com/youtube/v3/guides/auth/installed-apps
  * zsh, ‘&’ is special. Use single quotes around it to avoid errors.
  * scope must be a string char[1], separate multiple scopes by space

REF: https://stackoverflow.com/questions/53357741/how-to-perform-oauth-2-0-using-the-curl-cli#53357742et CLIENT_ID=Replace_with_your_Client_ID

### Youtube constants

```
auth_url=https://accounts.google.com/o/oauth2/v2/auth
token_url=https://oauth2.googleapis.com/token
base_url=https://www.googleapis.com/youtube/v3
uri_redirect=
# Per google docs, scopes are separated by whitespace
scope='https://www.googleapis.com/auth/youtube https://www.googleapis.com/auth/youtube.force-ssl'

client_id=$(Rscript -e "cat(Sys.getenv('OAUTH2_ID'))")
```

### Youtube Pagination

```
(in .tex, use math {)
part= snippet, content...

(study JSON)
fields=nextPageToken,items(id,snippet(title,description,publishedAt))
fields=nextPageToken,items(snippet(topLevelComment(snippet(videoId,textDisplay))))
fields=pageInfo.totalResults

####    Run this in neovim terminal, copy+paste into browser, which asks user
\begin{verbatim}
permission and then returns auth.code !
echo \
'https://accounts.google.com/o/oauth2/v2/auth?'\
'client_id='$client_id'&redirect_uri=urn:ietf:wg:oauth:2.0:oob'\
'&scope='$scope'&response_type=code'
\end{verbatim

####    We now have auth code.



PURPOSE:    HTTR2::  Given ONE video,  return ALL Comments
Authorization: Bearer [YOUR_ACCESS_TOKEN]
Accept: application/json


(1APR2022)
```

Google’s example, with loop for uri_redirect https://accounts.google.com/o/oauth2/v2/auth? scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fyoutube.readonly& response_type=code& state=security_token%3D138r5719ru3e1%26url%3Dhttps%3A%2F%2Foauth2.example.com%2Ftoken& redirect_uri=http%3A//127.0.0.1%3A9004& client_id=client_id

```
    -  Google's authorization server: https://accounts.google.com/o/oauth2/v2/auth

### Finally, Request:  appropriate query sent to:
GET https://www.googleapis.com/youtube/v3/commentThreads


### From Explorer
GET https://youtube.googleapis.com/youtube/v3/playlists?part=snippet%2CcontentDetails&maxResults=5&mine=true&key=[YOUR_API_KEY] HTTP/
```

```
Authorization: Bearer [YOUR_ACCESS_TOKEN]
Accept: application/json


###
same, but as Curl

curl \
  'https://youtube.googleapis.com/youtube/v3/playlists?part=snippet%2CcontentDetails&maxResults=5&mine=true&key=[YOUR_API_KEY]' \
  --header 'Authorization: Bearer [YOUR_ACCESS_TOKEN]' \
  --header 'Accept: application/json' \
  --compressed



###   From Google Playground
   https://youtube.googleapis.com/youtube/v3/commentThreads?videoId=Mec9sw1cJk8&part=snippet,replies
###

\newpage
#   CURL |  YOUTUBE API | GOOGLE API |  OAUTH 2.0 |

Examples:

##  cURL write (to standard)
##  w response after callling example.com


curl -w "Response %{response_code}\n" example.com


# github
curl https://api.github.com/zen

##  returns lot of kev=value pairs
curl https://api.github.com/users/defunkt

##   -include headers
curl -i https://api.github.com/users/defunkt

##   headers only
curl --head <URL>

##  CURL_CONFIG (a FILE)
##  USAGE   curl -K CURL_CONFIG ...


url = example.com
-w "Type:  Hello  %{local_ip} \n"
```

Misc Notes: “State” - cookies used to be used; now state carried in headers

Misc Notes: “State” - cookies used to be used; now state carried in headers

```
\newpage

Thanks for willing to take a look.
Some thoughts to get you and anyone else a start:

  -  I expect my issue connecting **httr2** and **google api** (youtube) to resolved with  simple parameter wrongly set.  As usual, i

  -  First, review vignette https://httr2.r-lib.org/articles/wrapping-apis.html, including oauth and github.
  -  Second, review the command line tool curl https://curl.se.
  -  Next, look through Google's API documentation + related:

    -  https://developers.google.com/youtube/v3/getting-started
    -  Try api requests in both Google Oauth2 Playground:
    https://developers.google.com/oauthplayground/
    -  And in Google API Explorer:
       https://developers.google.com/youtube/v3/docs/

I have done the above without errors.  But  httr2 code returns 404.
```

404. That’s an error.

The requested URL was not found on this server. That's all we know.

client = oauth_client(id= client_id, token_url = token_url, secret = client_secret, key = API_KEY, auth = "body", # header or body

```
    name = "youtube_ONE_video_ALL_comments")
```

req <- request("https://www.googleapis.com/youtube/v3/commentThreads?videoId=Mec9sw1cJk8&part=snippet,replies") %>% req_oauth_auth_code(client = client, auth_url = auth_url, token_params=scope[[1]])

resp <- req %>% req_perform()

```
Some Remarks:
  - Google is but one implementation of various API, oauth technologies.  The more you read the more confused you may become (at leas
  - The R package **gargle** is uses **httr** and therefore not my preference.
  - I am using httr2 to automate things;  I'd like to understand things using a little as possible:  curl, browser and local server r
  - Most of the R work is done at lower level, such as packages curl and httpuv.


#### HTTR2 - NOTES (needs clean up!)

PURPOSE:    Demonstrate configuration for HTTR2 and OAUTH2 with Google's Youtube API.

                            - uses off-the-shelf `httr2::req_oauth_auth_code()` + configuration
                          - uses authorization code flow.
                            - uses redirect_uri localhost, cut & paste (via obo) is deprecated.
                            - httr2::  hides almost all details of interaction.
                            - use  curl and localhost such as httpuv:: to see lower level

Source:  https://developers.google.com/youtube/v3/guides/auth/installed-apps

RELATED INFO:
  -  Google Explorer (youtube)
    -  Google OAUTH2 playground


#   ===============================
From Google (Youtube) Explorer:
GET https://youtube.googleapis.com/youtube/v3/playlists?part=snippet%2CcontentDetails&maxResults=5&mine=true&key=[YOUR_API_KEY] HTTP/

Authorization: Bearer [YOUR_ACCESS_TOKEN]
Accept: application/json
#   ===============================

For youtube (auth code):
```

echo "curl -Lsv "https://accounts.google.com/o/oauth2/v2/auth?
client_id=$client_id&
redirect_uri=https://127.0.0.1:8080&
scope=https://www.googleapis.com/auth/youtube&
response_type=code""

scope = list( "https://www.googleapis.com/auth/youtube", "https://www.googleapis.com/auth/youtube.force-ssl")

For youtube (obtain results): curl
'https://youtube.googleapis.com/youtube/v3/playlists?part=snippet%2CcontentDetails&maxResults=5&mine=true&key=[YOUR_API_KEY]'
–header 'Authorization: Bearer [YOUR_ACCESS_TOKEN]'
–header 'Accept: application/json'
–compressed

## NEEDED SCOPES:

https://www.googleapis.com/auth/youtube Manage your YouTube account https://www.googleapis.com/auth/youtube.force-ssl See, edit, and permanently delete your YouTube videos, ratings, comments and captions

playlistId = "PLlXfTHzgMRUIqYrutsFXCOmiqKUgOgGJ5" # Pavel Grinfeld, Linear Alg 3

#### Procedure:
  -     Follow hadley outlines in Vignette for Github and and getting user's information.  (Requires oauth token)
  -  Change for google
    -  let httr2 handle the details, use this function: httr2::req_oauth_auth_code()
  -  If I have this right, this will (1) get the access token and (2) complete REST
request.

# ==============================

# Mon May 16 01:40:07 PDT 2022

###  Index Cards

  -  LUA/language/neovim use
  -  GIT
  -  API/Curl
  -  R - debug, env, frames, roxygen2, ...
  -  SEARCH
  -  LINUX - mostly config
  -  VIM - daily use
  -  ZSH - daily use

LINUX on laptop
  -  Settings: selected linux
  -  downloaded and began, opens terminal:  Linux !
  -  Chrome, browser, data all seems in place !
  -  Only terminal is linux.

Source:  Rose Pesotta (HD6509.P47)
1881 - ass'n Alexander III
repression; ends period of limited reform
BUT seed planted during liberalization remains, now underground discussions, travelers, variety of ideas, esp in shetls.   Boys faced
1881 - 1914 1/3 of East Europian Jews go to US.

1760
George III (~ 17) educated, but poor understanding ppl.
Continent (esp France) respect English power, but not English culture, resistance to change, a Parliament that acquiases.   FRANCE is

=============================
###     ChromeBox:  Convert to Linux
=============================
  -  internal hard drive is /dev/sda, sandisk, 29.48G
  -  Chrome's partitions - do not mess, G- Chrome did a lot of things and is
        fussy.
  -  USB drive aka /dev/sdb 200+ GB

Developer Mode
ie code VERIFY is off.

Recovery Mode:  When you screwed it up; won't boot etc.
How to get:  must use internet; separate machine
Must be installed on bootable media (NOW:  SD thumbdrive)
This mode allows boot from USB/SD;  code is signed by Goolge; allows mode transitions.

Legacy Mode: Why called this?  Using legacy part of ROM? no G- support

Change from pure Chromebook to something else
Mr ChromeBox and Chrx DO WORK, with several gottchas.
Mr. Chromebox fixes up ROM, in one of 2 ways.  In my setup, partial ROM replacement; other way is FULL, but  I don't want to mess wit

Chrx is actually installs linux (on dev/sdb) but carefully not screwing Chrome's partitions on /dev/sda. Note:  installs to device /d
Chrx now gives you ^L (legacy) as well as ^D option
Chrx now gives you ^L (legacy) as well as ^D option

Both Mr. ChromeBox & Chrx can be run quickly.  When in doubt, no harm to reRUN.
**NOTE:   Chrx immediately destroys /dev/sdb partitions-- CAREFUL.**

To install linux, MUST boot to chrome (^D), get CLI, run chrx.
Do NOT install linux any other way (even if appears to work - use Chrx)
Do NOT think iso from SD drive will do it. Maybe; or not.

~~~~
I could only get GalliumOS to install; issues with Ubuntu 22.04 (wifi bug) and Ubuntu 20.04 did not work at all.
(SEE also wifi notes)


A lot of times things HANG;   just redo Mr Chromebox/Chrx (remain later destroys /dev/sdb)

LINUX on old Acer Laptop.
Simple:   In Chrome settings, turn on 'linux'   Chrome stays and terminal window get created.   Maybe best of both?   Simple, works.
(typed this in vim on laptop, in linux window)

Misc LINUX notes, details.
EFI - (partition) file format for execuatables, defacto standard for linux/BSD.


####    wifi
    Hopeless?  Ubuntu bug (May 2022) Some notes otherwise:
    -   EAP is protocol |  many pieces | goal:  protect wifi
    -   WPA several versions
    -   supplicant - one end seeks to be authenticated by other end.
    -   Standard is 802.1X

    - nmcli is main cli way. (see INDEX C)
    - networkctl status
    -   systemctl <command>
    - NOT an issue with GalliumOS (based on 18.04 ubuntu - so stuck here for
        now)

```vimdoc
This is block with 3 back ticks AND vimdoc:  boring!

Patience !   Takes a few minutes to finish.
shell 137 = out of memory

SOURCE FILE, for vimdoc:
~/code/jimHelp/source/jimHelp.md

CREATE:
jimHelp.txt in ~/code/jimHelp/doc/

PWD:
MUST be ~/code/jimHelp/

PANDOC:
!pandoc --metadata=project:xxx --lua-filter doc/panvimdoc/scripts/skip-blocks.lua --lua-filter doc/panvimdoc/scripts/include-files.lu


FINALLY,
:helptags ALL
~~~~

vim:nospell