Revision Notes: Minimal manuals

What are manuals (for)? Several types. But important type is to enable people to use the interface i.e. for supporting end-user action. This type is in effect part of the UI, an extra screen. Comparable to on-line help, help menus. There are other kinds of manual e.g. to look up what a command in a script does, how to maintain the system etc.

How manuals are used in practice. They are not read. Even if users say they like to read stuff, they don't for long. Why? One reason is how to understand what is being referred to.

Interaction of manual and machine: of (technical) language and what it refers to. To understand what the terms mean, you need to look at the machine (which is what you want to understand). This obviously applies to technical terms (zoom box), but also to "menu" (what are they like on this system?), to "mouse", and even to "paper tray". Such language only works in the presence, or vivid memory of, the thing itself.

Writing type (style). This goes against the basic approach to writing that we have all (painfully) learned from childhood: that in most writing, the difficulty is in describing everything that you (the writer) knows and can see to someone who isn't there. With a manual for supporting a practical task, however, it's the other way round: the machine IS there, and the writing is like speaking to someone present with you and the machine. You don't need to describe what is obvious and visible, and anyway you probably can't very well. People don't read but try things out, partly because the text isn't comprehensible without looking at the machine. You can't fight this; actually it's also an advantage, allowing great brevity. Thus this writing is NOT like most writing — letters, novels, etc. for making present something remote from the reader. It isn't an encyclopaedia for recording everything known about a device (although technicians may need this kind of manual).

How are manuals used? (ToA) They are used during, and to support, normal user tasks. Their design must therefore fit in with ToA in two ways:
1. The content of the manual will be the information missing from the naked UI but needed (at least by novice users) to perform each step of ToA.
2. The structure, form, visual nature of the manual must itself support action, each step of ToA, in the task that the manual is designed for: supplying just-in-time information.
Ordinary manuals fail at both: they are not designed to be *used*, so they are not usable, and so not used

Minimal manuals / Action centered manuals
The technique is commonly known as "Minimal Manuals" or "Minimalist Instruction" but better understood as "Action Centered Manuals"

Carroll J.M. (1990) The Nurnberg funnel: designing minimalist instruction for practical computer skill (Cambridge Mass: MIT press).

Draper S.W. & Oatley K. (1992) "Action Centered Manuals or Minimalist Instruction? Alternative theories for Carroll's minimal manuals" in P. Holt & N. Williams (eds.) Computers and Writing: state of the art (Intellect Books: Oxford).

Carroll J.M. (1998) (ed.) Minimalism beyond the Nurnberg funnel (MIT Press: Cambridge, Mass.)

Overall principle (level 1)
Action centered manual (not "minimal" except as a consequence)

Two aspects of "action centeredness" (level 2)
Organised to support doing tasks with the program
Organised to be used during action, as a resource

Three "super" principles (level 3)
1 Organise the design around user activity. (Principle of what to cover)

(Principles of what to leave out:) *Users act in a context of knowledge and information from sources other than the manual — they do not suspend disbelief and read manuals like fairy tales.*
2 Design around the effect of users' prior knowledge (and beliefs and desires).
3 Design around the effect of users' (future) knowledge during action.
(I.e. Exploit the information available from the interface at the relevant moment.)

Carroll's Design rules (level 4)

A. Focus on real tasks and activities

B. Slash the verbiage

C. Support error recognition and recovery

D. Guide exploration (Suggest tasks; suggest a good learning sequence)

E. Get the user started fast

F. Promote users' coordination of display and manual. (I.e. to use text as basis of observation and action, not as only a story to read.)

Components of a manual entry

The purpose or goal of the action. This tells the user what the action will accomplish, but it thus also tells them the reason for reading and using this item, and so acts as a title for the item and is the user's method for selecting the item from the manual. NOT usually the command name.

How to perform the action. These are the instructions for executing the action, though it can be almost blank if this is obvious to the user from screen prompts etc. It may mainly consist of aspects the user is most likely to neglect e.g. "don't forget to press <return>".

How to recognise success. Often this is obvious, but sometimes the user must learn (by being told) how to recognise the effects e.g. in Unix, no message means success; in some Macintosh applications, graph output may appear in a buried window and the user must be warned to search for this until it is uncovered.

How to recognise, diagnose, and recover from errors. This section is perhaps the greatest single source of the superiority of "minimalist" manuals. It is vital to supply information about error recovery *where* it is needed: immediately after the information about executing the action.

Other components. For instance some declarative explanation or "comment" e.g. knowing about the hidden "clipboard" buffer that most editors have. If the manual is intended to support learning by guided exploration, then a component listing a set of things to try may be useful.

User characteristics
(Most users will ....)

Only be motivated by real tasks;
 Can only make sense of the activity via real tasks

Jump the gun i.e. will get started fast anyway

Read the manual in any order

Use the situation i.e. pick up information from the screen, other users etc.

Are not always careful planners
Do not always follow instruction recipes systematically
Improvise things to do, and how to attempt them
Use and even seek errors in order to test boundaries and learn

BUT Are not good at recognising, diagnosing and recovering from errors

Reason and invent explanations, whether or not there is enough evidence. This is subject to interference from what they know about other, superficially similar, situations.

Most learners are domain experts

Carroll's way of deriving the principles from characteristics of human learning

The symbol "->" may be read as "because", and "=>" as "therefore". The general form is thus:
user error type ->
characteristic of human learning =>
minimalist instruction principle

People tend to jump the gun ->
People learn by doing; they try to act in order to learn =>
1. **Allow the user to get started fast**

People are not always careful planners ->
People learn by thinking and reasoning; they generate and test hypotheses in order to learn =>
2. **Rely on the user to think and improvise**

People are not good at systematically following instructional steps ->
People seek to work in a meaningful context and toward meaningful goals =>
3. **Direct training at real tasks**

People's reasoning about situations is often subject to interference from what they know about other, superficially similar situations (abduction) ->
People rely on their prior knowledge when they try to assimilate new experience =>
4. **Exploit what people already know**

People are often poor at recognizing, diagnosing, and recovering from errors they make ->
People use error diagnosis and recovery to explore the boundaries of what they know =>
5. **Support error recognition and recovery**

Major steps in creating a minimal manual

1 Design analysis: Observations on needs and problems existing prior to designing a manual

2 Subskill testing: Systematic testing of each bit of the manual against what you expect it to achieve. Basically this is systematic iterative design using think aloud protocols.

3 Criterion testing: Final experiment on the overall effect in controlled conditions.

What a manual should cover

What knowledge do the users have already

What is final target state of user skill.
Design a ref.man too?

Where (which manual?):
1) Articulatory material
2) Minimal core set of commands
3) Expanded set to give reasonable efficiency
4) The rest (quick ref.)

What to include:
Top level tasks
Subgoals (~ system operations in central arena)
Articulatory subgoals (e.g. how to invoke a menu command)
General error recovery procedures
General information finding procedures

Exercise: write a minimal manual

The (only) way to understand this topic properly is to write your own minimal manual, and of course test it out on at least a few users. To keep this tractable, pick a really small topic, but aim to have your test users succeed immediately on the device. Suitable topics might be:

- Using your mobile phone (your target users: people who have never used that model of phone, or a mobile phone at all). If that's too easy for users, perhaps sending a text message, or calling a number you have stored. People want to use a phone instantly: you are aiming for no errors, no hesitations, even.

- Doing a stats procedure in minitab your users haven't done before (multi-variable ANOVA? ...)

- Finding a room in the psychology building (for people who have never been there); or in some other building. The issue here is: what instructions are the best complement to the clues visible in the building? A map gives hugely non-minimal information....

- Instructions on using a photocopier — perhaps for a non-elementary task e.g. double sided printing, stapling and collation — for someone unfamiliar with that particular machine.

- Or pick you own (best of all).

Example page from a Minimal manual

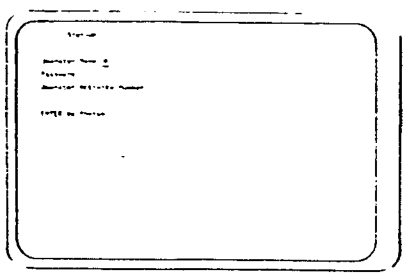
QUITTING WORK

When you leave the computer unattended (e.g., at the end of the day) you need to tell it that you are quitting.

You want to "Sign-Off".

You need to get the Task Selection menu to do this.

If you signed off correctely you should see the following display on the screen:



The computer is now ready for someone to use.

If you cannot get the Task Selection menu, try

or

Be sure you have typed the correct abbreviation for "Sign-Off".

Example Page from a Minimal Manual for Unix editor "vi"

Basic Editing

Starting up

vi myfile <return> will start up the editor on the file "myfile".
This will work even if "myfile" does not yet exist (you may want to create it).
Try: vi /etc/printcap You can modify this in the editor, but you won't be able to overwrite it.

Moving the cursor

Use the arrow keys if you have them. But they can cause problems on Macs.
The keys **'h' 'j' 'k' 'l'** also work as arrow keys.

Scrolling

If you try to move the cursor off the bottom or top of the screen, the text will scroll to keep it in view.
Try: moving the cursor to the last line of the file. N.B. you cannot move it beyond the last line.

Insert text

i This will enter insert mode: any text you then type will be inserted until you type **<esc>**
Try: inserting letters in the middle of a line;
inserting a fresh line (go to start of a line, type text, **<return><esc>**).

Deleting

x This will delete the character under the cursor
J (N.B. Capital **J**, not 'j') This will join two lines, thus in effect deleting the linefeed between them.
? If it inserts 'x' or 'J' instead of executing them, press **<esc>**, delete the 'x' or 'J', try again.

Saving the file

:w<return> will write the new version of the file to the existing file name, overwriting old version.
? If it inserts ':w' instead of treating it as a command, press **<esc>**, delete the ':w', and try again.

Leaving the editor = quit = exit

:q<return> will leave the editor.
? If you get an error message on the bottom line, the command failed.
? If you see the shell prompt, the command succeeded.

:q!<return> abandon your changes and leave the editor without overwriting the original file.

Tasks to try

1. Try each of the tasks above, including the suggestions marked " **Try**" above. These use a standard system file (/etc/printcap) which you can modify in the editor, but which you won't be able to overwrite.

2. Create and edit a file which you can send as an electronic mail message to yourself:
   vi msg
   [Now insert and edit text to suit yourself]
   mail $USER < msg [This will send it to you by e-mail.]