How to make a CD

M. Schwab [1]

/usr/local/src/our/cdrom/docprod/paper.tex

ABSTRACT

blablabla

Introduction

SEP's CD-ROMs facilitate the technology transfer of SEP's research results to its sponsors and the preservation of these results for SEP's graduating students. Such a CD-ROM includes a standalone software collection: when mounted to a UNIX computer with X-window, C- and FORTRAN compiler, the CD-ROM allows a user (reader) to view the research results, to interact with interactive documents (e.g. through pushbuttons or editing of files), and to destroy the result and to recompute it from the source files enclosed on the CD-ROM.

To facilitate such an interactive, reproducible document environment, the CD-ROM comprises two classes of files: document files and system files. Document files are files which pertain to the document the CD-ROM stores. All document files have to be replaced when a new CD-ROM with a different document such as the next SEP report or thesis is produced. System files are the files which facilitate the interactive, reproducible access to the document. These system files comprise the essential computer software needed to present the research results. The system files have to stay up-to-date with regard of the SEP computer environment to ensure that documents developed and tested by the researcher on his workstation also work when exported on the CD-ROM.

The production of a CD-ROM at SEP comprises four consecutive steps:

- **Prepare:** Check the original document, and isolate the testing computer from the network.

[1] **email:** matt@sep.stanford.edu

- **Assemble:** Assemble and up-date the system and document files.
- **Test:** Create a disk image of the assembled software and test the document.
- **Finish:** Create a tape image of the assembled software, print the cover leaflets of the CD-ROM, save the production log files, and integrate the testing computer in the local network.

The Assemble and Test steps are iterated until the results of the tests are satisfactory and the assembled software can be copied to tape and send to the CD-ROM manufacturer.

PRODUCTION INTERFACE

Each of the four production steps – prepare, assemble, test, finish – comprises a set of small subtasks which are each performed by a single shell script. The prefix of the shell name indicates to which of the four production steps the shell belongs: `p, a, t, f`. I will leave the actual shells undocumented since they are subject to change and should be straightforward to read (most of them only consist of a dozen command lines). An interested reader should look at the individual shell script to find all the details he may want to know. I will rather describe the purpose of each shell script and how the person producing a CD-ROM - a cdmaker - is effectively using them.

The set of individual production shells are organized through a set of master shells: `cdmaster, cdconfigure, cdproduction`:

- The **cdconfigure** file lists all the variables shared by the individual files. It is the only shell file which a cdmaker has to edit when starting to produce a new CD-ROM. All the frequently changing variables (such as the cdrom name, the document list, the location of the original document) are placed at the top of the file.
- The **cdproduction** file lists all the small individual tasks performed by individual shells. The tasks are grouped into the four major production steps: prepare, assemble, test, finish. A cdmaker edits the cdproduction shell by setting the variable of a particular task, e.g. `p_waste`, to `yes` and others to `no`. The tasks corresponding to a variable set to `yes` is then executed when the cdproduction shell is run. (In detail, the cdproduction shell lists the tasks it is set to perform, requests a confirmation from the cdmaker, concatenates the cdconfigure shell and the selected task shell `p_waste.csh`, and executes the resulting script.) I often execute the shells of an entire production step such as *prepare* or *assemble* in one sweep by setting all the corresponding variables to `yes`. The variable tp_waste defined in the same line as `p_waste` contains a short description of the shell's purpose. This description is needed to document the production in a log file.

- Finally, The **cdmaster** shell executes the cdproduction shell. It copies the output of the file into a log file and thereby ensures a complete documentation of the production process.

If you wanted to add another shell to the list of cdproduction shells, you would have to write the actual shell (I suggest you implement the standard echo statements I use in all shells; these statements help reading the long production log files. Additionally, the shell should receive all its variable definitions from the `cdconfigure` file). In the `cdproduction` shell you would have to add a toggle variable `p_newjob` set to `no` and a description variable `tp_newjob` with a short purpose statement. The prefix of the variable should reflect the group (`prepare, assemble, test, finish`) the shell belongs to. Finally, you have to add the toggle variable to the list of toggle variables in the `cdproduction` shell.

All individual shell scripts create an individual log file named `../log/shellname.log` which helps for analyzing problems and mailing authors. An additional logfile, `../log/prodlog`, contains all output from `cdmaster` during a CD-ROM production.

In summary, a cdmaker starts a cdproduction by editing the variables in the cdconfigure shell. Later during the production, he only edits the cdproduction shell by setting the variables corresponding to the tasks he wants to perform. He executes these tasks by invoking the cdmaster shell which in turn calls cdproduction and stores a log of the output of the cdproduction shell.

THE FOUR PRODUCTION TASKS

Overview

When producing the CD-ROM, the cdmaker copies the software needed on the CD-ROM from the local computer environment to an assembly area. In order to test if the assembled software is functional when exported on a CD-ROM, a test computer is isolated from the network and the assembled software is installed and tested on it. The cdmaker resolves any detected problems by correcting the assembled software. Once the cdmaker finds the test results acceptable, he makes a final copy of the assembled software to tape which is sent to the outside CD-ROM manufacturer.

This sequence of steps can be represented by a loop of four major steps, (`Prepare, Assemble, Test, Finish`), as described by Figure 1. Each of the steps is accomplished by set of small- to medium-sized shell scripts. The next four paragraphs list the four major steps and the corresponding shell scripts.

Prepare

- `p_isolate` isolates a computer from the local network. The computer does not have access to any local SEP software or directory. When testing the CD-

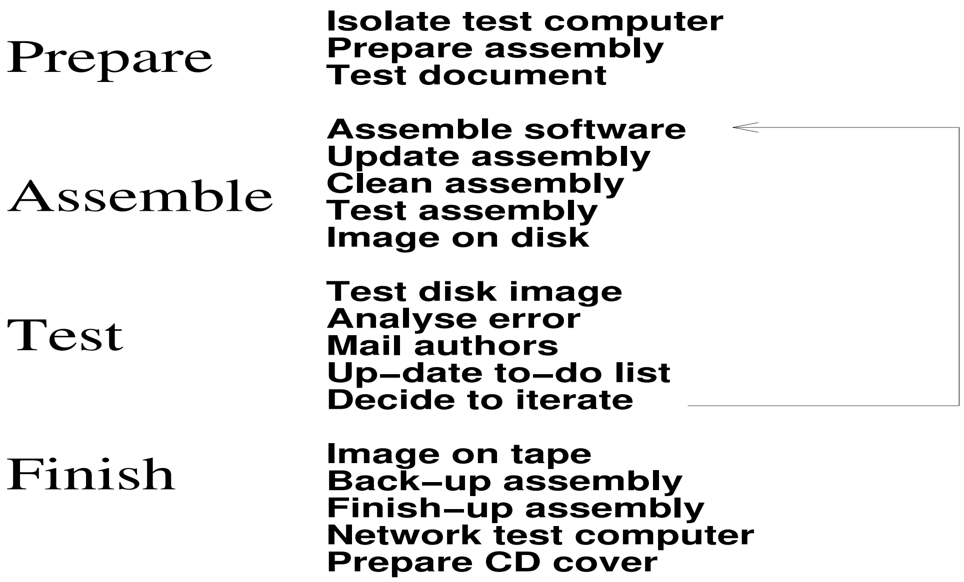

Figure 1: Flowchart of the four major CD-ROM production steps. The arrow indicates the iteration of the assembly and test step until the assembled software passes the tests. The middle column illustrates the major steps by listing the essential tasks each step comprises. $\boxed{\text{list}}$ [NR]

ROM on this computer the cdmaker effectively ensures that the software is independent of SEP's local environment.

- `p_size` measures the size of the documents which will be copied to the assembly area. The cdmaker can make sure that the document does not exceed the size of the assembly area. If it does exceed the assembly area, the next few shell scripts allow the cdmaker to detect junk files within the document.
- `p_waste` applies Wastestat to the document (see selfdoc).
- `p_execstat` applies Execstat to the document (see selfdoc).
- `p_figstat` applies Figstat to the document (see selfdoc).

Assemble

Next the cdmaker copies all the needed software to an assembly area. Often the copying also changes certain variable settings which were specific to SEP's local environment. All the copying is done by scripts which automatically set all variables to values required for export of the software.

The copying is not always necessary: if the assembly area contains software from a recent CD-ROM production, the cdmaker may prefer to up-date only certain parts of the assembled and tested software. The cdmaker however has to make sure that the software in SEP's environment and on the CD-ROM do not differ to a point where software developed at SEP may not execute properly when joined with the software on the CD-ROM[2]. If the assembled software is up-dated instead of copied, the cdmaker has to run the scripts `a_update` and `a_grepper` to up-date all variables referring to the previous CD-ROM.

The following list of shell scripts copy software packages from SEP's computer system to the assembly area:

- `a_mkdir` creates a directory structure for the assembly area. This shell has not been used in years and is only needed when the cdmaker decides to assemble a CD-ROM from scratch.
- `a_cake` copies cake. The shell copies the cake source code, the executable, and the SEP cake rules. This should be up-dated frequently since cake rules change quickly and authors rely on them heavily.
- `a_make` copies make. Similar to `a_cake` except it deals with gmake. For a transition time we will have to provide both, cake and gmake.
- `a_tex` copies the tex stuff.
- `a_xtex` copies the xtex stuff.
- `a_youngm` copies some files of Young Minds, such as `cdlink` and `cdcopy` which are needed when building a linktree from a CD-ROM or copying the entire contents of a CD-ROM.
- `a_seplib` copies seplib. This should probably be up-dated on every CD-ROM since it changes rather frequently and it is one of our main products.
- `a_xtpanel` copies the xtpanel stuff.
- `a_shells` copies a bunch of scripts which cdmaker uses for testing the CD-ROM, such as `Tour` and `Figstat`.
- `a_utils` copies all the software which did not fit into any of the previous categories (e.g. `perl`).

The following shells copy documents to the assembly area:

- `a_idocs` removes old documents from the assembly area and copies new ones. `cdconfigure` lists the documents which are removed and copied. To replace an old version of a document with a new one, the cdmaker just places the document in the list of old documents and new documents. Documents which remain unchanged are not listed in either list.

[2] I suggest to up-date every major software package once every half year. All documents and `seplib` I copy for every CD-ROM.

- `a_cdroot` copies all the files for the CD-ROM base directory, such as a `README` file or installation script. Some of these files list the documents contained on the CD-ROM: again this information is listed in the `cdconfigure` file.

Another set of shells cleans the assembled software:

- `a_chown` changes the owner of all assembled files to matt. This shell will prompt you for the root password.
- `a_chmod` sets the permission of all files to read- and writeable by everybody.
- `a_update` finds all occurences of old CD-ROM and document variables and replaces them with an up-dated variable. The old and new variables are defined in `cdconfigure` (e.g. `OLDCD` versus `CDNO`).
- `a_dvi` regenerates the dvi file for all documents. Currently, I do not use this shell since `bibextract` is broken.
- `a_lib` converts archives to random libraries which can be linked faster.
- `a_clean` removes junk files from the assembly. It searches for certain file names which indicate junk files, such as `junk.old`. Additionally, it executes a `Tour clean` in all documents which triggers the clean rules in each chapter.
- `a_waste` suggests additional files for removal (See selfdoc of `Wastestat`).
- `a_execstat` finds executables in the document directories (see selfdoc of `Execstat`). The documents should not contain any executables. Executables are unwieldy since they tend to be large and machine specific. Instead the documents include the source code corresponding to a described process. The compilation of the source code is an important part of the reproducibility of the documents results.
- `a_grepper` searches each file for the occurences of old version numbers. I run it sparingly since it takes long and `a_update` seems to be fully debugged by now.
- `a_size` computes the size of the assembled software. The maximum space available on a CD-ROM is 673 MBytes. The company which wrote the software to make disk images encourages us to limit the size of a CD-ROM to 650 MBytes. When creating a CD-ROM file system, the Young Minds software adds tabulating files in each directory which accounts for about 30 MBytes. Therefore the assembled software should ideally be 620 MBytes but should not exceed 640 MBytes.

Test

The last paragraph described how the cdmaker assembles all the needed software for a CD-ROM production. Next the cdmaker has to test if the assembled software is functional when exported on a CD-ROM. Since producing a CD-ROM is time consuming and costly, the cdmaker simulates a read-only CD-ROM by making an image of the CD-ROM onto a disk. In this process the disk is loaded with the assembled software encoded in the format the final CD-ROM will be encoded in (ISO 9660). The disk's UNIX file system table is erased and after making the image, the disk behaves exactly like a CD-ROM.

Having created a CD-ROM image on disk, the cdmaker tests the image by detecting junk files, inspecting the documents and destroying and recreating all figures.

These tests should be done on different platforms by cross-mounting the disk image.

- `t_image` images the CD-ROM onto a disk. This step requires a return stroke by cdmaker to confirm the imaging process after the file system depth and size has been computed. Check that the file system is not too deep (8 layers of directories) and that the size does not exceed 650 (673) MBytes.
- `t_browse` executes the shell `browse` in the CR-ROM root directory. It brings up the documents without building a linktree (SUN only).
- `t_tree` builds the linktree. This shell is interactive and executes the `installcd` shell of the CD-ROM root directory[3].
- `t_grepper` searches each file for the occurences of old version numbers. I run it sparingly since it takes long and `a_update` seems to be fully debugged and `Grepper` was probably run on the assembly area before.
- `a_waste` finds junk files in the disk image (see selfdoc of `Wastestat`).
- `t_execstat` finds executablesin the documents of the disk image (see selfdoc of `Execstat`).
- `t_figstat` finds junk figures in all documents of the disk image and tabulates all figures (see selfdoc of `Figstat`).
- `t_clean` cleans all the documents by executing `Tour clean` (see selfdoc `Tour`.
- `t_burn` removes all the reproducible figures of all the documents of the disk image. Than it lists the existing figures by calling `Figstat`. It should not list any reproducible figures.
- `t_burnCR` Same as burn but removes the conditionally reproducible figures.
- `t_burnburn` Same as burn but removes the reproducible and conditionally reproducible figures.

[3]This shell should be executed in the background using a script or *expect* to feed its interactive input.

- `t_figures` up-dates all figures which are out-of-date or missing (`Tour figures`). Then it lists the existing figures (`Tour figstat`). All figures listed in the `Cakefile's FIGLIST` definition should exist.

- `t_cycle` visits all documents and removes each reproducible figure and recreates it. This shell simulates a reader who burns and rebuilds every reproducible figure which he encounters when browsing through the document. Testing an entire document using `t_cycle` rather than `t_burn, t_figures` requires more time since after each figure reconstruction the directory is cleaned: shared resources between figures are rebuilt from scratch for every figure (`Tour cycle`). However, `t_cycle` simulates a reader's behavious more closely.

The errors found when testing the disk image have to be corrected. If an error is due to defects of the assembled software, the cdmaker has to obtain and copy a correct version of the software. The results from testing the documents are communicated to the authors of the document. This should be done in a semi-automatic fashion using the shell's individual log files in `../log`. The authors than correct the original document. Finally the cdmaker up-dates the assembly area by copying up-dated or corrected files using the assembly shell scripts.

If the cdmaker is satisfied with the quality of the assembled software and its disk image, he proceeds to finish the CD-ROM production by creating an CD-ROM image of the assembled software onto tape which is sent to the CD-ROM manufacturer.

Finish

In the last step of the CD-ROM production, the cdmaker prepares a tape for the CD-ROM manufacturer. Additionally, he makes a back-up copy of the assembly area and saves the log files. Finally, the test computer is integrated in SEP's local computer network.

- `f_image` images the assembly area onto tape. This step equals the step to make disk image. Again the operator needs to confirm the command by hitting the *return* key after the copy software checked the directory tree. The cdmaker has to make sure that a tape is inserted in Oas' tape drive before issuing this command. Additionally, he needs to write the size of the CD-ROM software on the tape's label. This size is stated by the copy software before asking to confirm the copy command (The precise size is needed by the CD-ROM manufacturer).

- `f_backup` copies the assembly area to the Oas' tape drive. The cdmaker is again responsible to put a tape into the tape drive before copying the software.

- `f_protocol` moves the log file of the CD-ROM production to a log directory. The shell copies two log sets of log files: the file written by `cdmaster` and stored in `../log` and the files written by Young Mind software and stored in `/usr/local/src/their/cd-tools`.

- `f_network` formats the test disk `testcd` as a standard UNIX disk, copies the `fstab` and `rc.local` files, and reboots the test computer in a networked mode. This shell will again interactively prompt for the superuser password.

The disk when sent out to the sponsors is accompanied by a short description pamphlet. The cdmaker needs to copy an old version of this pamphlet in `our/cdrom/diskcover`, edit it, and print a paper copy.

Additionally, the CD-ROM is adorned by a print on the top surface. The two postscript files for this print (`sepdisk.dmfg.ps, sepdisk.text.sepcd9.ps`) are stored in `our/cdrom/diskart`. The cdmaker needs to copy an old version of the postscript text file (e.g. `sepdisk.text.sepcd9.ps`), read it into `idraw`, and edit it. To check the new versions, I suggest to print out the two postscript files. If the layout of the files is satisfactory, the cdmaker *fetches* the two files to the terminal Macintosh and stores them on a floppy disk.

Finally, Diane needs the cover pamphlet copy, the floppy disk, and the CD-ROM image tape to arrange the production of the CD-ROM by a manufacturer.

ONE-OFF CD-ROM

At SEP we usually produce a single one-off before producing the entire edition of a CD-ROM. This one-off is tested the way a disk image is tested (described above).

Ideally, the disk image should be tested on all platforms (I need to make the `testcd` accessible to all our platform). Currently, the disk image is thoroughly tested on our test computer (a SUN). The one-off CD-ROM is used to test the CD-ROM on the other platforms. It is often not possible to isolate a computer from the network temporarily to test a CD-ROM. In these cases, I mount the CD-ROM and set the environment variable PATH to exclude any SEP specific directories. I need to improve the testing on different platforms.

APPENDIX PROBLEMS

- installcd: does not copy correctly on SUNs: cp -r for rockridge standard
- installcd: call by another script
- check if INSTRUCTIONS up-to-date
- bibextract: does not extract references correctly
- bib files: always include `SEP.bib, Misc.bib`
- document figstat, wastestat, execstat, down-sizing, reproduction time list of executables on CD-ROM (`docdocpaper.tex`)
- data sets: scan for absolute pathnames

- sephelp responsibility list
- assembled utilities: up-dates more frequently, get expert approval (right way to copy `xtpanel`?)
- bios: include automatically only the needed bios
- organization software: functional packages: all tex stuff etc
- tex fonts: overcome xtex problem (good description)
- machine names: cdconfigure variables (e.g. *zarand* `testmach`)
- automail: individual shell files split by author, shell specific message