The Humboldt County Election Transparency Project and TEVS

by Mitch Trachtenberg

June 29, 2013

Development of this document was funded by a grant to Humboldt County from the United States Election Assistance Commission.

Note: Contents page references and pagination modifications added by Dick Atlee, 21 Jan 2016
PDF available at http://dickatlee.com/issues/elections/evote/pdfs/EAC_Grant_TEVS_Docs_1.pdf

1. Overview

This is a guide to the process used by the Humboldt County Election Transparency Project (the ETP), and a set of instructions on how to operate the open source TEVS vote counting software.

The ETP is a citizen-based volunteer effort to independently scan all ballots cast in Humboldt County elections. The scans are then made available to any interested party. In this way, people can have access to ballot images generated by an independent group, even though they cannot have access to the actual paper ballots that have been cast.

The ETP has satisfied its goals when it has created and signed a collection of ballot scans that is made available to the public.

There are both advantages and disadvantages to the ETP approach. One disadvantage is the risk that the ballot images do not accurately represent the ballots. In our project, we rely on our ability to trust one another to not falsify images -- in a situation where there is no independence on the part of those doing the scanning, trust may become an issue.

We recommend spot checking image files against the paper ballots after a transparency group has committed to the accuracy of its image files by digitally signing (*) the collection. If your group believes that spot-checking is insufficient protection against fraudulent images, you could use a risk-limiting audit to confirm that the images are “faithful enough” to the paper ballots.

(*) One process for generating a digital signature is described in section 9.

Another disadvantage to this technique is that it relies on being provided with a complete set of all cast ballots in unaltered form. In reality, the only way that this requirement can be completely satisfied is if a transparency project has access to the cast ballots prior to their movement from whatever point they enter the official election system. This would require independent scanning of ballots directly at precincts, as well as independent scanning of ballots at any locations where absentee and/or mailed ballots are opened and processed. The logistics of this are beyond the ETP’s capabilities, and we therefore rely on the legitimacy of the official chain-of-custody documents provided by the Humboldt County Elections Department. Note that this logistical compromise is also a necessity for any procedure that relies on the integrity of the paper ballot collection itself.

The advantage of the ETP approach is that it supplements the black box of election tabulating equipment with a more transparent independent scan. Rather than rely on the numbers reported solely by proprietary tabulating software, the images of the ballots themselves become available to anyone wishing to count votes from them. In this way, all citizens can gain access to the images that the official tabulation equipment reduces to numbers. Misreads by the official

tabulation equipment can be exposed. Attempts to fraudulently alter ballots become potentially detectable, because the ballots can be viewed by an unlimited number of citizen inspectors, rather than by a limited set of auditors. It becomes possible to catch simple mistakes which would otherwise go undetected, because without a second count, there is no way of knowing when and where the first count might incorporate errors.

Another advantage to the ETP is that it brings a set of independent outsiders into the election process. It is valuable for an elections office to establish a reputation as being open to having outsiders examine their processes. Elections officials are rightly concerned about the security issues of having large numbers of outsiders handling ballots -- the ETP represents a rational compromise between the security concerns of elections officials and the transparency desires of the public at large.

Once the ETP has generated a collection of ballot image scans, the scans are made available to anyone who might want to count votes off of them by any technique anyone might choose. People can manually count votes using the images or use their own software to analyze the images for votes.

By offering ETP scans to candidates, elections offices can enable candidates to investigate results themselves prior to making a decision to request an expensive official recount. This dynamic has already been seen in Humboldt County.

A volunteer with the ETP has developed vote counting software called TEVS, and TEVS is used in Humboldt County to generate independent vote counts. These counts can then be compared with the official counts generated by the County's official equipment. TEVS is open source and licensed under the GNU General Public License, meaning that it can be freely duplicated without charge and that every line of computer code run by TEVS may be examined by anyone. TEVS runs on the open source Linux operating system, which can also be freely duplicated without charge. TEVS is proof that such software can be developed -- there is no doubt that organizations with greater resources can and should develop similar but better software.

2. Chain of custody

"Chain of custody" is a paper trail recording who has had evidence in their possession, when they obtained the evidence, and when they surrendered it. If the signatures making up a chain of custody are valid, and if the people who have signed have actually treated the evidence as they attest in signing, the evidence can be trusted to have not been altered. (The chain of custody does not prove that the evidence was not altered -- it only indicates who has had possession of the evidence, and therefore, potentially, the ability to have made alterations.)

Even if the elections office does not require you to, you will want to extend whatever chain-of-custody documentation existed up to the point you've received access to the ballots. That means that when you open any sealed container, there should be two people present, and you

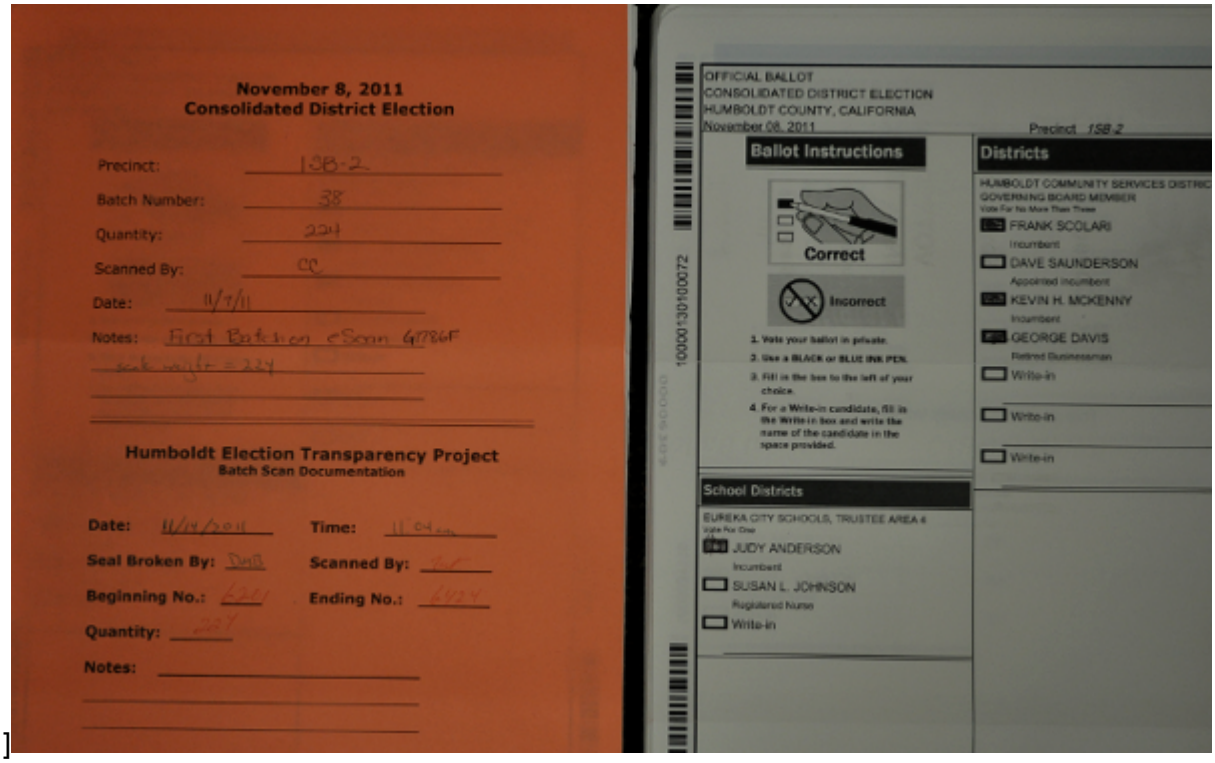
should fill out a form indicating the time and date at which the container was opened, and the person doing the opening. (You can note any witnesses as well.) Similarly, you should fill out a form with the time the container is re-sealed, and who did the sealing. The only people who should have access to the scanning area at the time a container is open are those who are directly involved with the count or scanning operation.

It is best to open only one or two ballot containers at a time, deal with their contents, and then reseal the containers.

[Photo 1. Ballot batches are retrieved and returned by County personnel to this locked ballot storage room. Entry and exit from the room is logged in the room.]

[Photo 2. A sealed ballot bag as received from the County.]

[Photo 3. A ballot custody form from 2011. Material on the top of the form records information about the official, County scan. Material on the bottom records information about the later transparency project scan. Subsequent versions of this form also record the date and time that ballot batches are resealed, and the initials of the person resealing the batch.


November 8, 2011
Consolidated District Election

Precinct: 13B-2
Batch Number: 38
Quantity: 224
Scanned By: CC
Date: 11/7/11
Notes: First Batch on eScan 47864
sack weight = 224

Humboldt Election Transparency Project
Batch Scan Documentation

Date: 11/14/2011 Time: 11:04 am
Seal Broken By: DnB Scanned By: JJ
Beginning No.: 6221 Ending No.: 6224
Quantity: 224
Notes:

OFFICIAL BALLOT
CONSOLIDATED DISTRICT ELECTION
HUMBOLDT COUNTY, CALIFORNIA
November 08, 2011
Precinct 13B-2

Ballot Instructions
Correct
Incorrect
1. Vote your ballot in private.
2. Use a BLACK or BLUE INK PEN.
3. Fill in the box to the left of your choice.
4. For a Write-in candidate, fill in the Write-in box and write the name of the candidate in the space provided.

Districts
HUMBOLDT COMMUNITY SERVICES DISTRICT
GOVERNING BOARD MEMBER
FRANK SCOLARI
Incumbent
DAVE SAUNDERSON
Appointed Incumbent
KEVIN H. MCKENNY
Incumbent
GEORGE DAVIS
Retired Businessman
Write-in
Write-in
Write-in

School Districts
EUREKA CITY SCHOOLS, TRUSTEE AREA 4
JUDY ANDERSON
Incumbent
SUSAN L. JOHNSON
Registered Nurse
Write-in

10000130100072

]

Photo 3, in addition to showing a chain-of-custody form, shows a randomly selected ballot from

the corresponding batch. The ballot has already been scanned and numbered. The number is on the left center, very near the left edge, beneath the much blacker and somewhat larger ballot type number.

Although this is a digital photograph, it could in theory be compared against the digital scans provided by the project, as additional evidence that the project's released images had not been altered.

You may wish to maintain continuous video surveillance of the scanning area; the ETP makes no such effort.

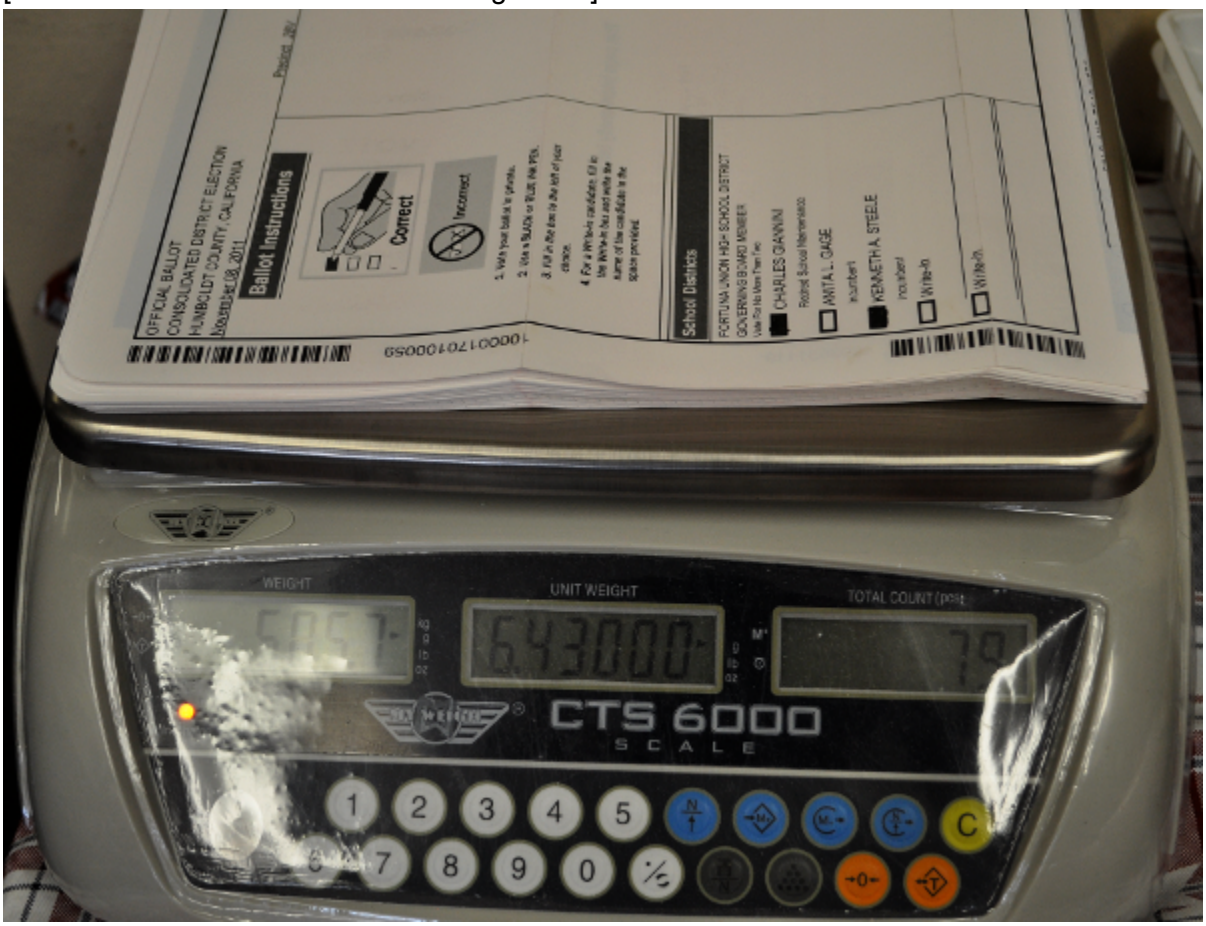
3. Marking implements

Only a non-standard color of pen or pencil should be available in whatever area is used for scanning. Forms should be filled out using that non-standard color. For example, if ballots are typically marked in black or blue, green or red might be the only color used on ETP forms.

4. Counting ballots to be scanned

Ballot containers coming to the ETP are marked with the number of ballots enclosed. The ETP uses a digital counting scale (CTS 6000) loaned by a member to count the ballots in each container prior to scanning them. This scale is easily available online for approximately $200 and is commonly used for weighing groups of small parts (nuts or bolts, for example). It can be connected to a computer, though the ETP does not do so. The scale has been accurate for groups of up to approximately 100 ballots; larger groups can be split to measure as batches of less than 100.

[Photo 4. Ballots on CTS 6000 counting scale.]



The scale is straightforward to operate. A set of 10 ballots is placed on the scale at the beginning of a session, the scale is put in initialization mode by pressing the “parts stack” key, and the quantity 10 is entered. The “parts stack” key is pressed a second time. The scale calculates the weight per ballot and is then able to read out the number of ballots in later batches.

The number of ballots reported by the scale is written down alongside the official count, and any discrepancies are noted.

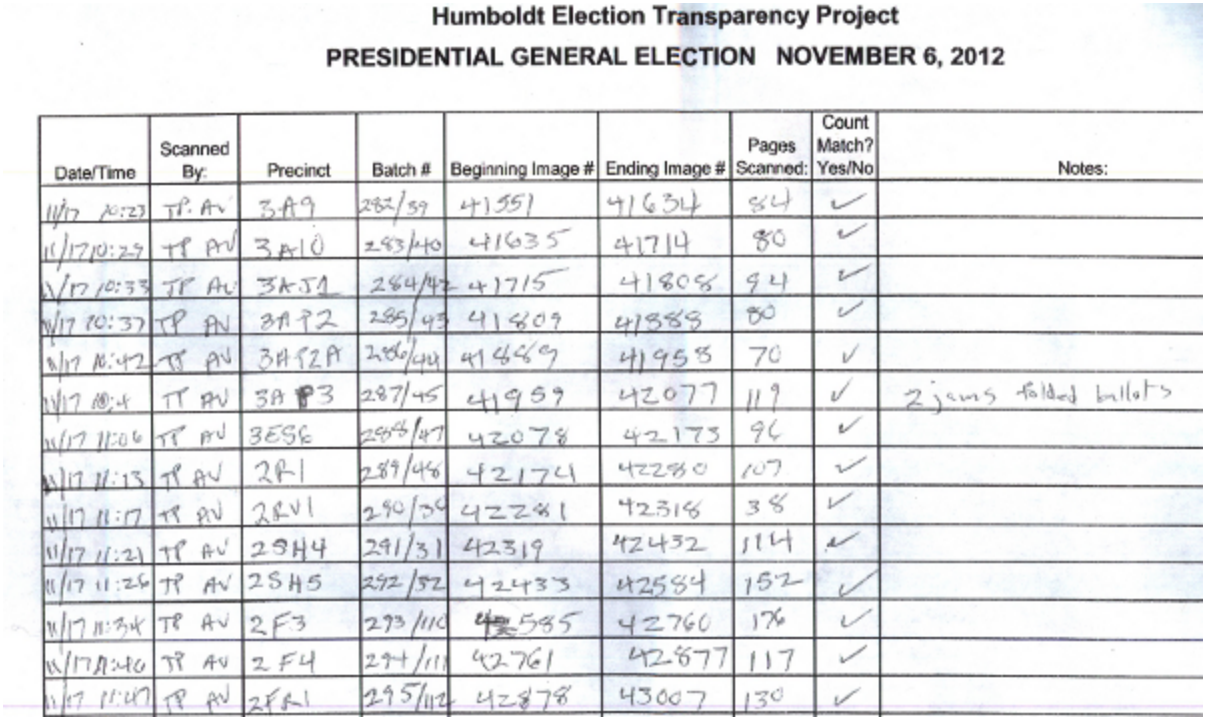
5. Scanning

The ballots are then scanned, using a scanner which imprints a serial number on each sheet of paper. The scanned images are saved in computer files which are numbered with the corresponding serial numbers. The starting and ending serial numbers of each batch of ballots are recorded on a manual paper log, and the resulting count is also written down. If, for example, the calculated number of scans is one lower than the count from the scale and the count from the County, the assumption is that a double-feed has taken place.

A page from the scanning log used in the November 2012 election is below: the ETP records the date, time, and initials of the person performing the scan, the precinct of the batch being scanned, the County and transparency batch numbers (County batches are not brought up in order), the starting and ending serial numbers, the page count as calculated from these numbers, confirmation that this count matches the scale count and the County count, and any notes about the batch.

Much of this information can also be captured by the scanning software; however, we have chosen to write it on paper logs as well.

[Photo 5. Scanning log.]



Humboldt Election Transparency Project

PRESIDENTIAL GENERAL ELECTION NOVEMBER 6, 2012

Date/Time	Scanned By:	Precinct	Batch #	Beginning Image #	Ending Image #	Pages Scanned:	Count Match? Yes/No	Notes:
11/17 10:23	TP AV	3A9	282/39	41551	41634	84	✓	
11/17 10:29	TP AV	3A10	283/40	41635	41714	80	✓	
11/17 10:33	TP AV	3AJ1	284/42	41715	41808	94	✓	
11/17 10:37	TP AV	3AP2	285/43	41809	41888	80	✓	
11/17 10:42	TP AV	3AP2A	286/44	41889	41958	70	✓	
11/17 10:4	TP AV	3A P3	287/45	41959	42077	119	✓	2 jams folded ballots
11/17 11:06	TP AV	3ESE	288/47	42078	42173	96	✓	
11/17 11:13	TP AV	2R1	289/46	42174	42280	107	✓	
11/17 11:17	TP AV	2RV1	290/34	42281	42318	38	✓	
11/17 11:21	TP AV	2SH4	291/31	42319	42432	114	✓	
11/17 11:26	TP AV	2SH5	292/32	42433	42584	152	✓	
11/17 11:34	TP AV	2F3	293/110	42585	42760	176	✓	
11/17 11:46	TP AV	2F4	294/111	42761	42877	117	✓	
11/17 11:47	TP AV	2FR1	295/112	42878	43007	130	✓	

The ETP uses a Fujitsu fi-5900 scanner equipped with a pre-imprinter, which uses an ink-jet printhead to place a number on each sheet of paper as it enters the scanner. Less expensive scanners tend to have post-imprinters rather than pre-imprinters. Pre-imprinting is desirable, because it means that the image of the ballot will include the serial number, as printed on the ballot. However, post-imprinting still allows ballot image file numbers to be used to locate the source ballot. (Of course, this requires that the ballot image file numbers correspond to the serial number imprinted on each sheet. There is no reason to change an image's filename, though, so this is not a difficult requirement.)

[Photo 6. Fujitsu production scanner, model FI-5900C]

The Fujitsu scanner imprinters are very similar to ink jet printers, and use the same sort of ink jet cartridge. However, the ink jet cartridge is not automatically moved across the width of the paper as the paper goes by, creating lines of text. Instead, the cartridge remains in a set position, while the paper advances past it. This limits the region of the ballot which could potentially be affected by the ink to a narrow vertical channel. This channel should be chosen so that it cannot interfere with the reading of either the ballot information or the serial numbers.

The imprinters can be programmed to delay their printing by a varying amount of time after the paper has begun to pass by, enabling the number to be shifted higher and lower within the fixed channel.

Prior to scanning, it can be helpful to use a print shop's mechanical "paper jogger" to vibrate a batch of ballots into a neat stack. Such devices are regularly available for $200 on sites like ebay.

Double feeds

It is not uncommon for scanners to pick up more than one sheet of paper at a time. Scanners can be programmed to stop when they encounter a double feed but, especially when pre-imprinters are involved, the double feed may be detected only after the imprint has been made. (If such a situation occurs, the ink-jet print head can be manually and temporarily shifted when the ballot is re-run, so that both imprints are visible.)

For a double-sided ballot which was double-fed, the result will be images of one side of one ballot and the opposite side of the tag-along ballot.

The most foolproof way to ensure that no double-feed has passed by is to compare the available ballot counts with the count generated from the starting and ending serial numbers imprinted on the ballots. If the scanner's count has not increased sufficiently, one or more ballots have double-fed. In that situation, someone must hunt through the ballot stack, looking for ballots that did not receive imprints. For single-sided ballots, the remaining unimprinted ballots can simply be run through the scanner. For double-sided ballots, however, sides of both the unimprinted ballot and the preceding or subsequent ballot will be missing. The most straightforward approach is to delete the partial ballot image files, note the deletions in a log, and then rescan both ballots which had not been completely imaged. The partial images may be kept, but this leaves the risk that they will be mistakenly included in any counting that takes place. That is why deletion is preferable, along with a logged explanation of the resulting gap in the serial number sequence.

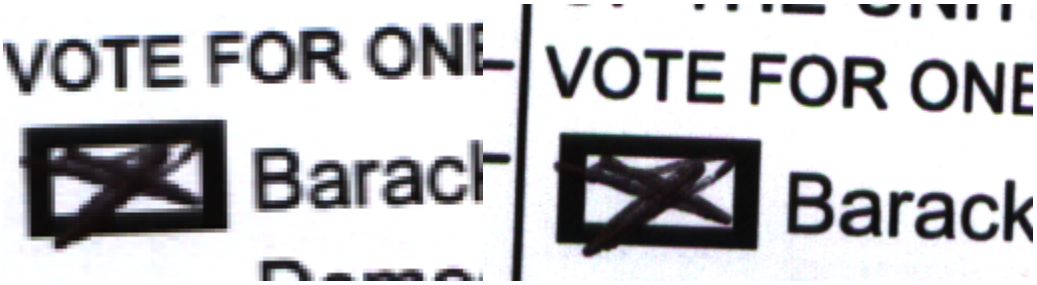
Miscellaneous scanning issues

With document scanning, there is a tradeoff between resolution and time, and also tradeoffs between image quality and file size.

Resolution and mode

To allow a human to determine which positions have received votes, a resolution of 100 dots per inch should be more than sufficient. The ETP typically scans ballots at 150 dots per inch, in full color. Higher resolutions are available, but transmission of the data from the scanner to the computer takes a long time unless the scanner itself compresses the image data prior to transmitting it.

[Photo 7: Scan at 150 dpi. Photo 8: Scan fragment at 300 dpi.]



The higher the resolution of an image, the more accuracy there will be in operations such as optical character recognition off the image. For this reason, 300 dot per inch scans of a blank ballot of each precinct or vote-by-mail code can be useful. For the bulk of the ballots, however, we have found 150 dpi to be more than sufficient.

In addition to grayscale and color modes, scanners may have a mode called “bitonal” or “black and white.” Bitonal or black and white images are a poor choice for elections purposes, as light gray marks easily seen on the ballots may be lost from the scanned images.

Time

Our throughput using the fi-5900c and open-source scanning software has been approximately 1,000 ballots per hour. This is far lower than the specified throughput of the scanner, which operates at more than 100 ballots per minute when using its hardware compression. The largest impacts on the time required is not the scanning speed, however, but the time to load and unload the scanner, the time required to handle the double feeds, and the inevitable time that the scanner is not actually scanning.

I’ve also used the smaller, lighter and less expensive fi-4120c scanner outside of Humboldt. (The current model closest to the fi-4120c is the fi-6140c.) The fi-4120c scanner will process 20 ballots per minute, leading to an anticipated throughput of more than 1000 sheets per hour. The actual observed throughput was 400 sheets per hour.

Although it is easy to understand why people will anticipate getting greater throughput than reported here, I’d strongly advise anyone to perform an actual test, closely following the chain-of-custody requirements, to determine what throughput they can expect in actual usage.

File size and quality

File size is less significant of an issue, but may still matter. Typically, our images are in the

vicinity of 100,000 (10^5) to 200,000 bytes per side. A terabyte sized drive, now easily available for under $300, can hold 10^12 bytes of data, meaning approximately 10^7 such images, or 10 million images.

JPEG compression does not retain all the information from the original file -- it is called “lossy” as opposed to “lossless.” You would not want to introduce compression artifacts into your images by over-compressing them. At 100,000 bytes per image, experts will spot artifacts, but these should not have any impact on the ability of humans or programs to determine which vote areas have been marked.

Housekeeping

We scan images into folders of 1,000 images each. This minimizes delays that might otherwise be caused by things like the automatic generation of thumbnails when a folder is opened. In most cases, ten or more such folders can fit on a single DVD.

Behavior and decorum

All those coming into contact with ballots are “sworn” by County staff. It is understood that the transparency project is non-partisan. Were we a larger group, we might make an effort to ensure representatives from both major parties were in the scanning room whenever ballots are open.

6. The TEVS Software

TEVS purpose is to count the number of vote markings for each choice for each position, and report how many marks each choice receives. In order to do this, it needs to be able to locate the choices and contests. Most existing vote counting software does this with the use of a “template” or “layout” file, that specifies the vertical and horizontal offset of each choice. TEVS also uses template files for this purpose. Under certain circumstances, however, it can generate the templates on its own.

To do this template generation requires 300 dpi resolution images and, unfortunately, in many situations, scanning all ballots at 300 dpi or higher can be time-prohibitive.

A compromise approach is therefore to scan one copy of each ballot style at 300 dpi resolution, run TEVS once on those ballots to generate templates, and then scan the remainder of the ballots at 150 dpi. This adds an unfortunate need for two sessions, but it is probably the most straightforward way of making use of the automatic template generation when available.

TEVS can do the scanning as well as the counting, but you can use whatever software you wish to generate the scan images, as long as you then arrange the images in the folders that TEVS expects.

TEVS is not yet “production quality” software and may never be. My primary purpose for TEVS

was and is to enable the motivated users of the transparency project to generate our own count of Humboldt County elections. Others may, however, find TEVS a useful starting point, as it demonstrates one way of using existing open source tools to conduct automated election vote counts.

Obtaining TEVS

A DVD containing Ubuntu Linux with the TEVS software, its prerequisite software, and the set of ballot images from a small Humboldt County election can be obtained by contacting the County of Humboldt Elections Department. A nominal fee may be charged for DVD reproduction.

Validating your copy of TEVS

The MD5 hash for the file used to generate the described DVD is:
MD5(201209162100.iso)= 14fb6979058bda67286dc3c32b837abe

Depending on your situation, for later versions of TEVS or similar software you may want to confirm that the copy of TEVS you have received has not been altered since it was digitally signed. While the TEVS disk includes software that can confirm a file against a digital signature, it is logically foolish to rely on a piece of software itself to tell you that it is valid. ("Are you a liar?" "No." "But what if you're lying now?" "I'm not." "Oh, OK.")

Therefore, you should use digital signature software to confirm that the digital signature that comes with TEVS confirms that its source is the source you expect. The digital signature relies on a private key known only to the person making the copy of TEVS; you validate it by providing a matching "public key." This "public key" can be retrieved from a variety of public key repositories, for example, MIT hosts a repository at http://pgp.mit.edu. You can search this repository for "Mitch Trachtenberg TEVS," or, if you are looking for a copy from someone who has modified the software, you can search for the information they provide.

(It's important to understand that, without further validation, you cannot know that the person who has created "Mitch Trachtenberg TEVS" signature is actually Mitch Trachtenberg. However, you can confirm two things: (1) the software is identical to the software signed by the person knowing the private key associated with "Mitch Trachtenberg TEVS," and (2) the "fingerprint" of the public key found at the repository matches the "fingerprint" advertised as correct by "Mitch Trachtenberg TEVS." You must have some way of confirming that the location offering the fingerprint is not defrauding you, and the best way to accomplish that may involve using the phone or the physical mail service to establish contact in a way less subject to attack than the internet.)

7. Installing TEVS

If you are uncomfortable with the idea of damaging your Windows installation, you may wish to physically remove your existing Windows hard drive from a computer and purchase a hard drive

for exploration. Hard disks are quite inexpensive now, and a drive of the necessary size can be found for less than $100. Any computer store technician should be able to remove your existing drive and install a new one in less than ten minutes, though you may need to convince them that you really do not wish to have your operating system transferred to the new drive.

Installation of the DVD containing Ubuntu Linux with TEVS preinstalled is identical to installing a standard Ubuntu DVD. TEVS is distributed attached to an operating system called Linux, running in a polished distribution called Ubuntu. If you have received TEVS on a DVD or external USB hard drive or flash drive, you will need to "boot" your computer off of TEVS rather than off of your computer's usual files. For many current computers, this involves holding down an "F" function key -- generally along the top of the keyboard -- when the computer is first turned on.
The exact procedure will vary from model to model, but you may be asked, for example, to "Press F12 for Menu."

A good description of the install process can be found here:
http://www.ubuntu.com/download/desktop/install-desktop-long-term-support

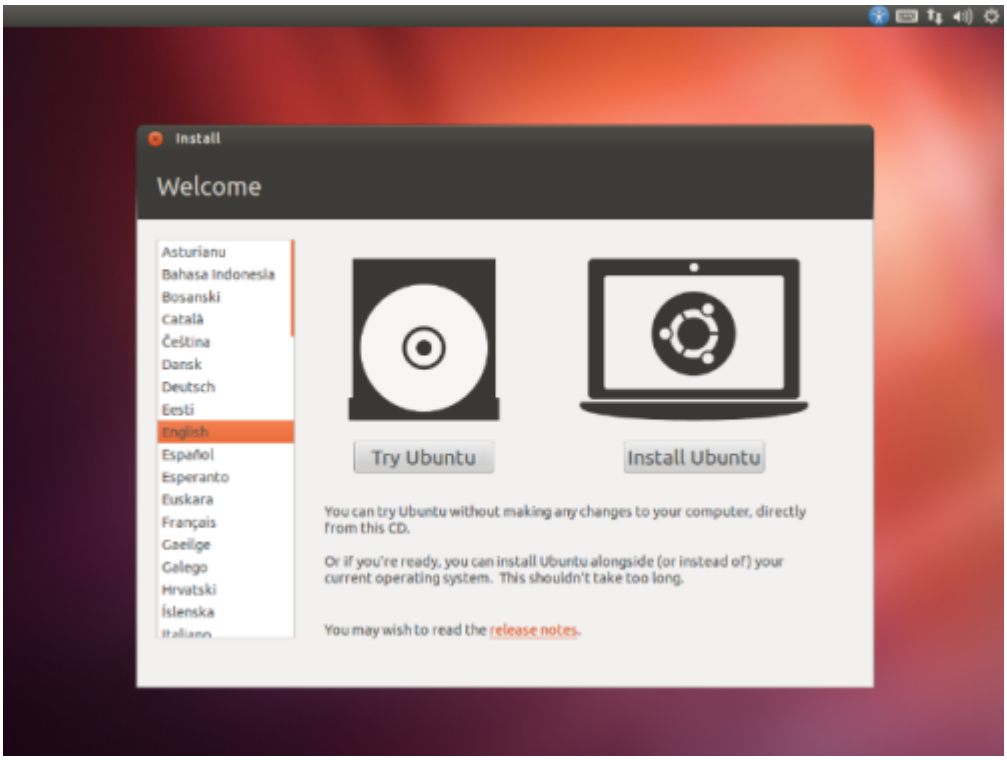
If you are using a computer which will be dedicated to TEVS, you should either install TEVS over the preexisting software on your computer's normal hard disk, or you should remove your computer's normal hard disk. It is possible for the Linux operating system to pull in files from any device attached to your computer -- removing the other devices can ensure that this will not happen. (TEVS will not do this on its own, so if you have established that your version is valid, you don't strictly need to detach the other devices from your computer. It can't hurt, though.)

Warning: recent computers designed to work with Windows 8 may make installation of open source operating systems needlessly difficult. It might be best to install onto a machine designed for Windows XP.

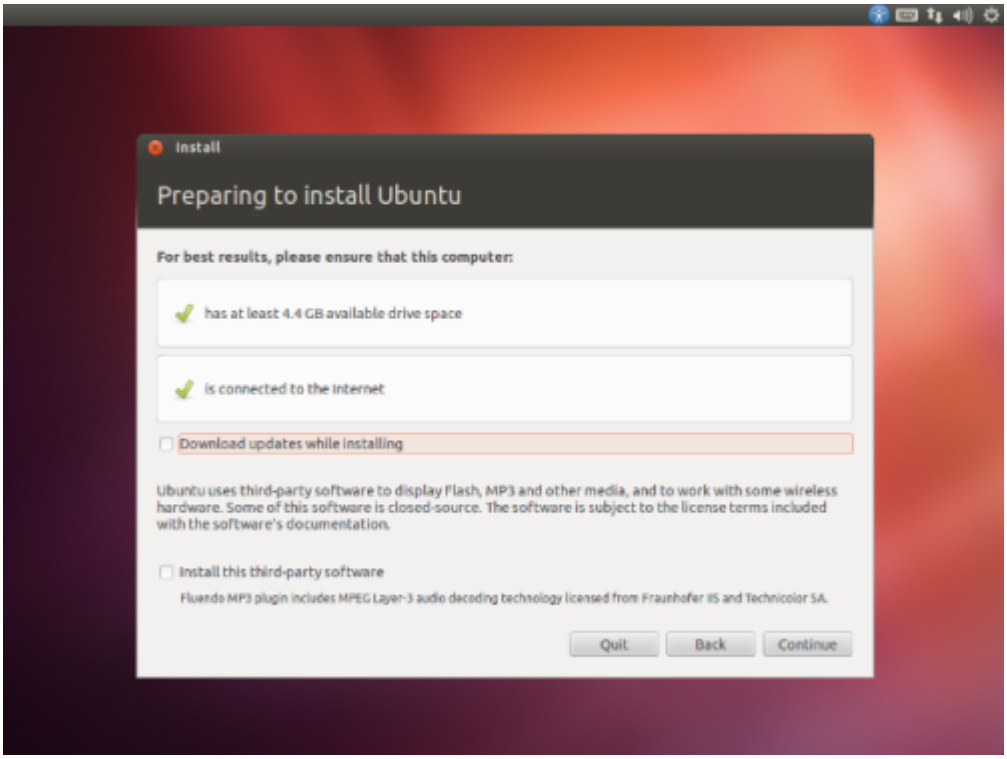
Warning: although the installation asks that you connect to the Internet in order to enable automatic updating, you would obviously not wish to connect to the internet were you using the software in actual service.

Because the idea of "installing an operating system" might sound intimidating, the following is a complete set of photographs of the screens presented during the process.

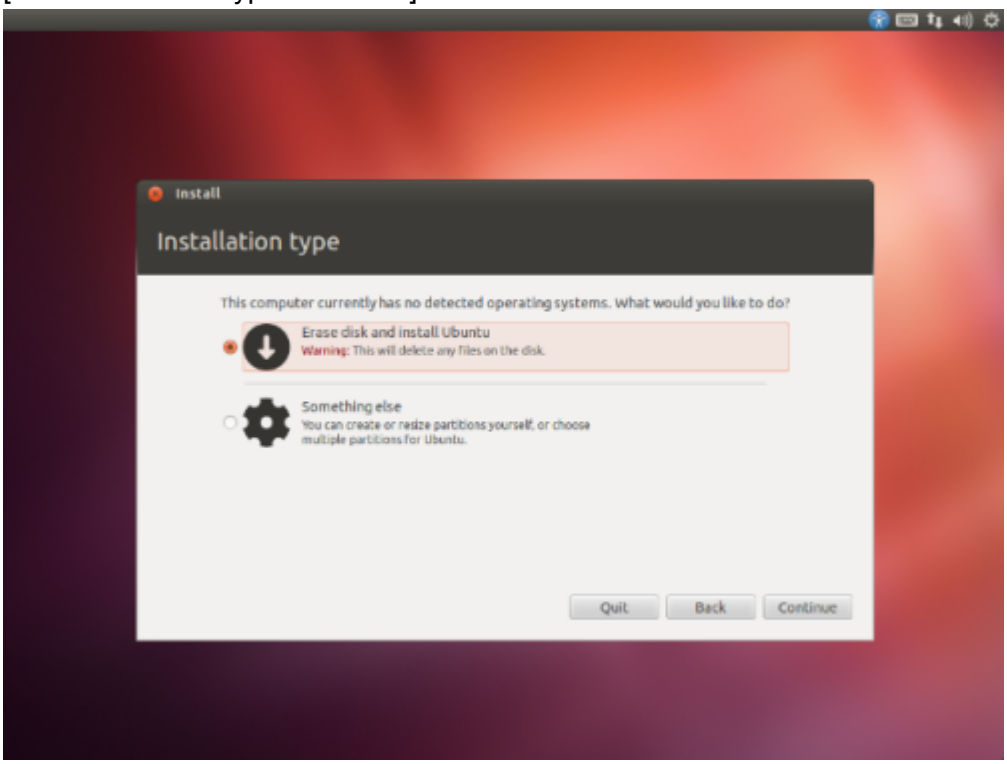
[Photo 9. Initial install screen for Ubuntu 12.04. Click on “Install Ubuntu”.]



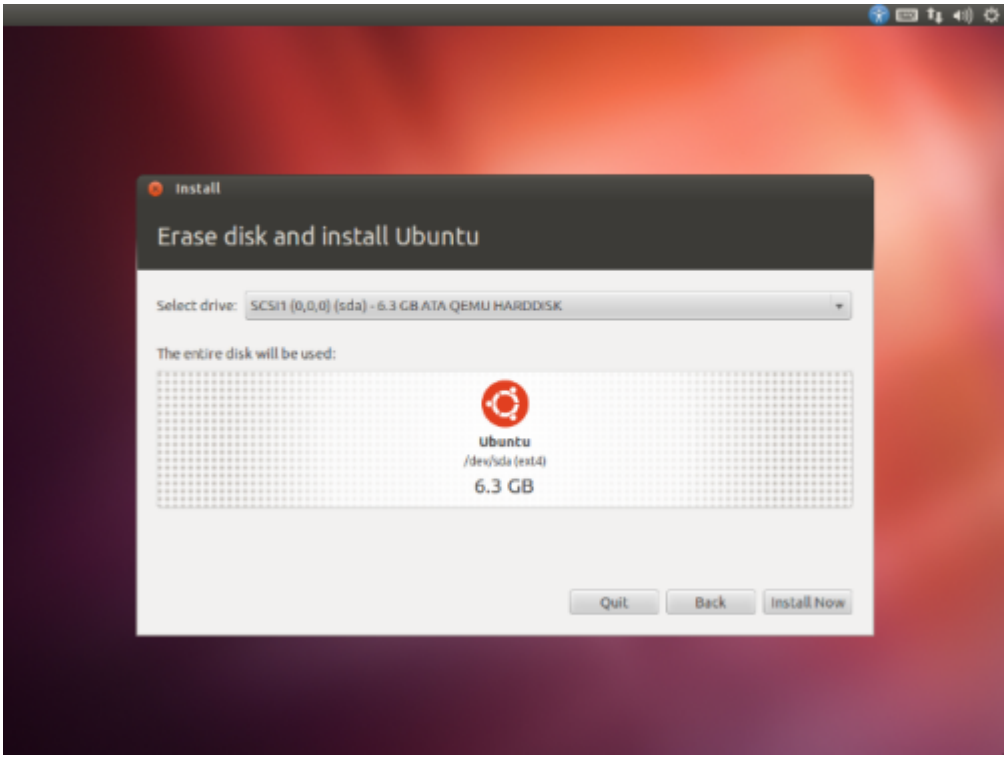
[Photo 10. Install screen.]



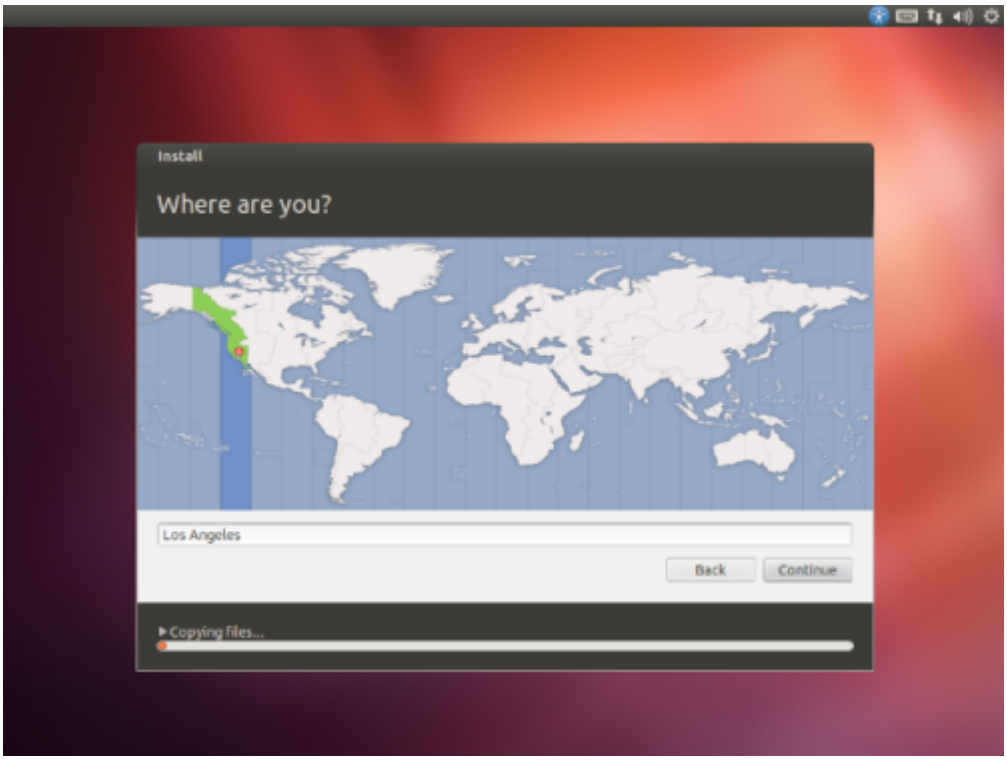
[Photo 11. Install type selection.]



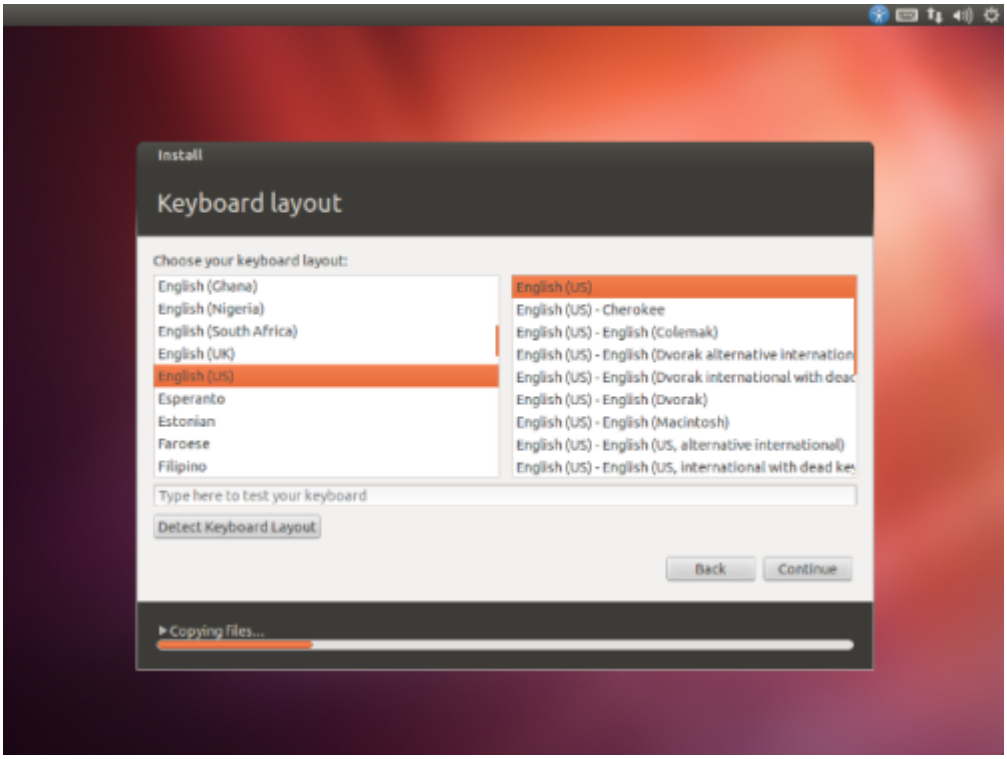
[Photo 12. Confirmation screen for disk erasure.]



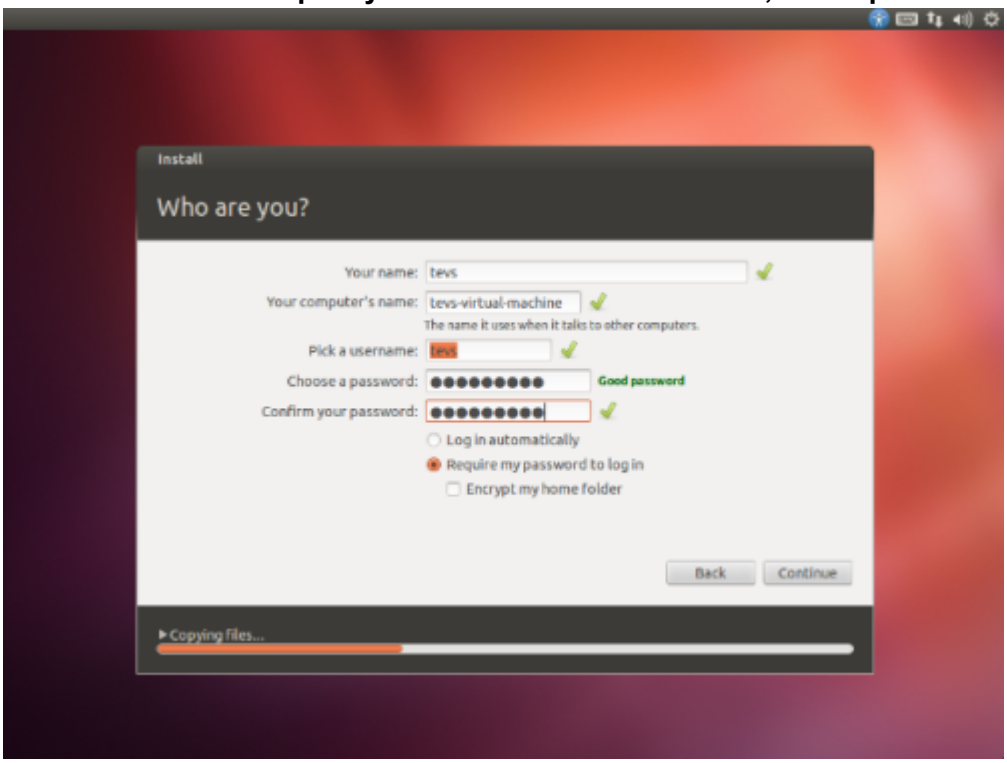
[Photo 13. Location screen.]



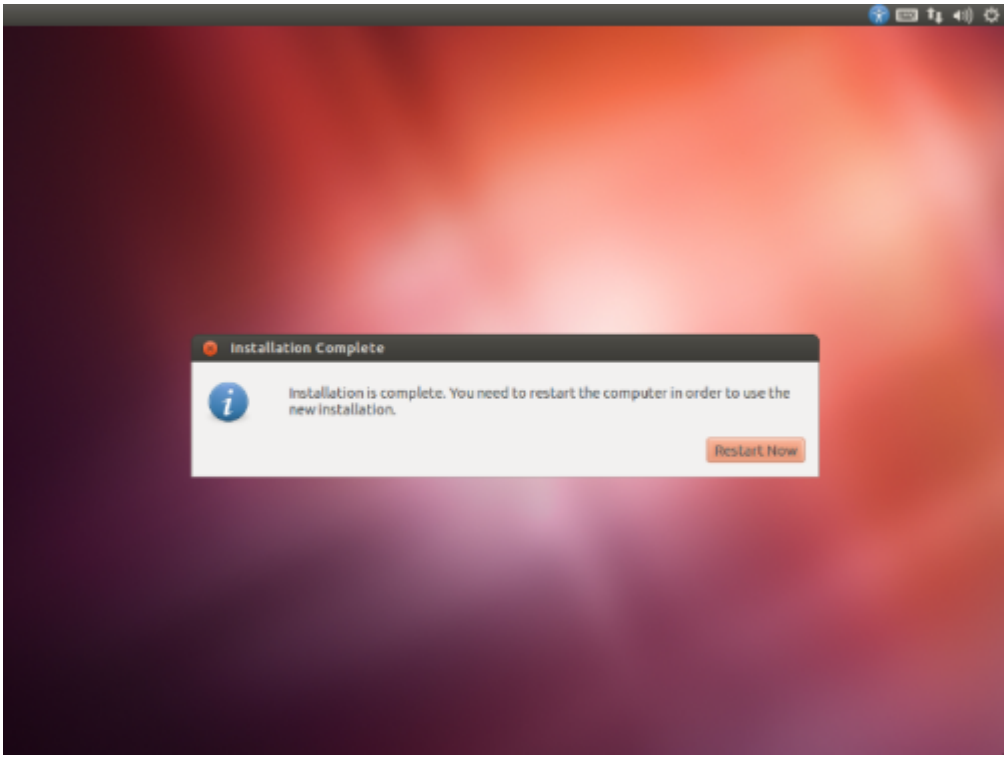
[Photo 14. Keyboard selection screen.]



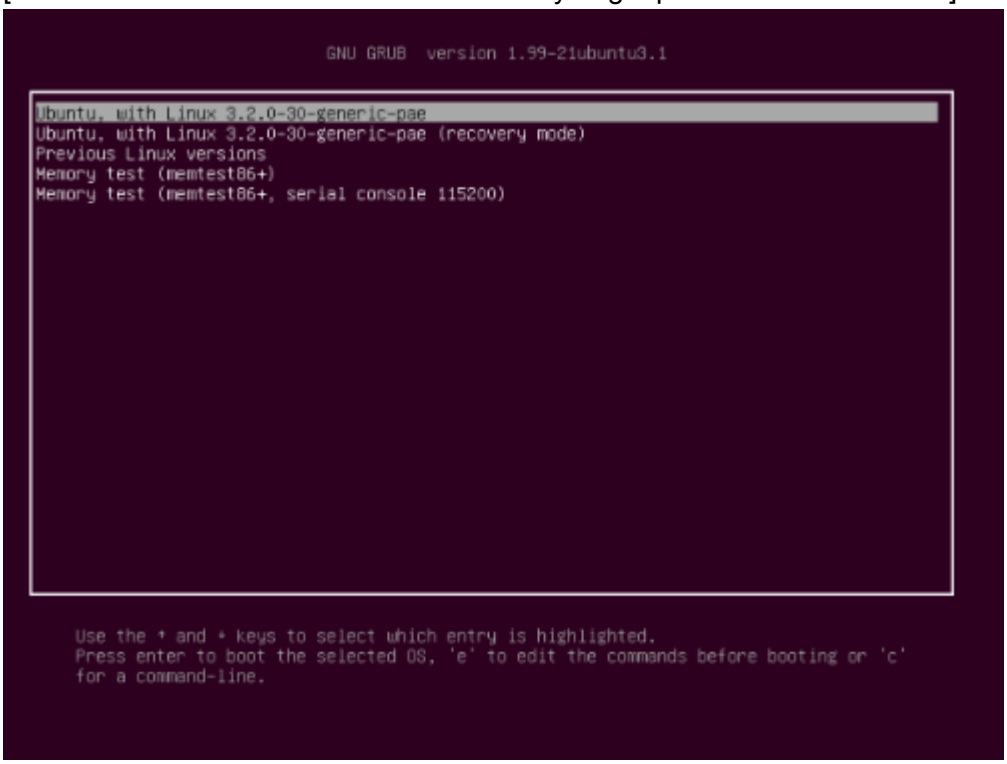
[Photo 15. Initial user name and password screen.
VERY IMPORTANT: Specify tevs as the initial username, or scripts will fail.]



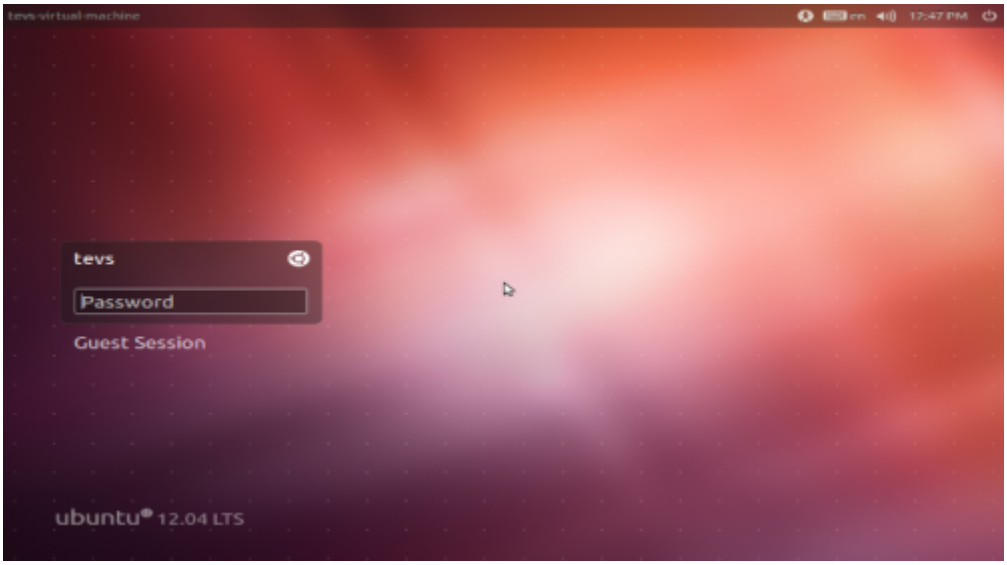
[Photo 16. Restart screen.]



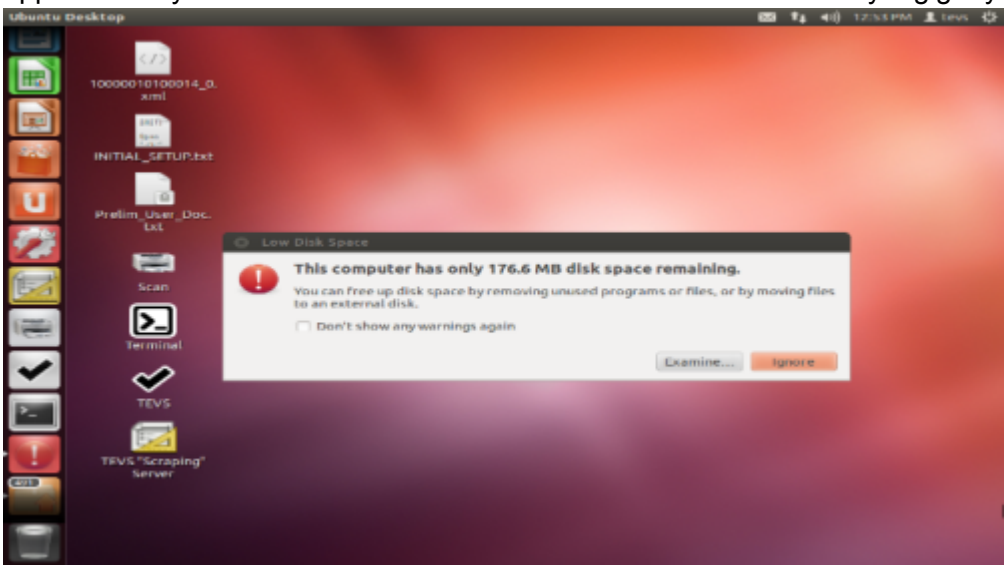
[Photo 17. After restart. Press the Enter key to get past this “boot” screen.]



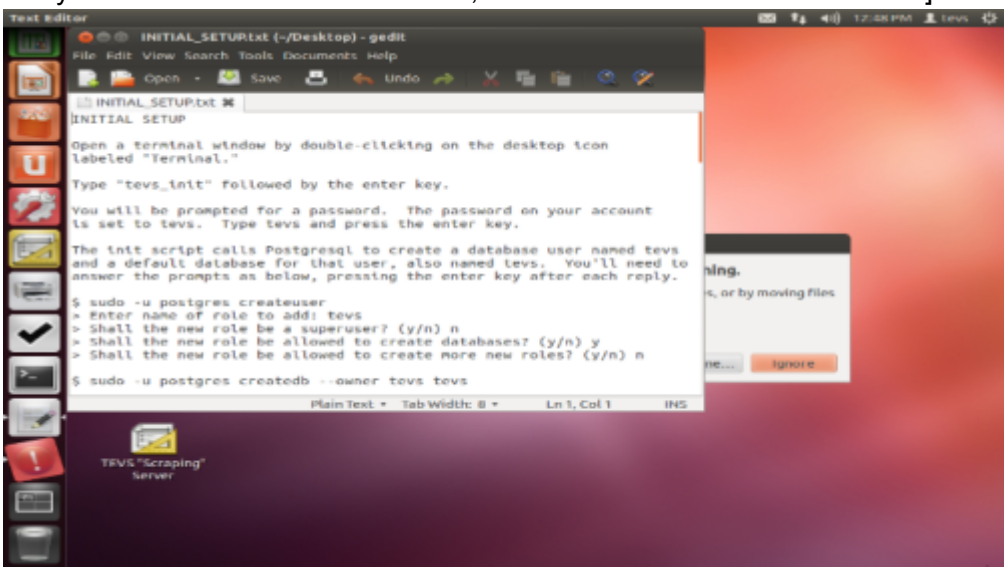
[Photo 18. User name and password prompt.]



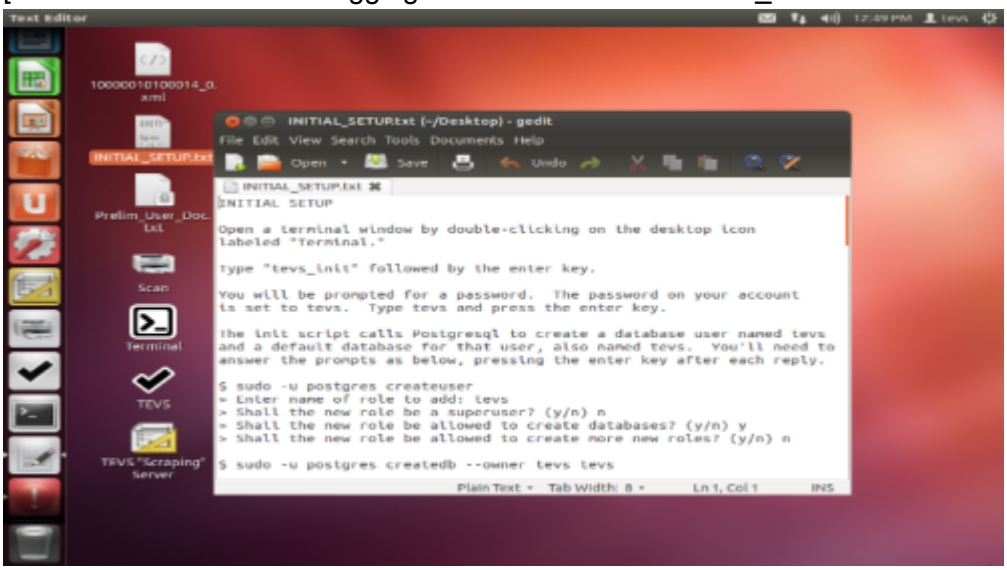
[Photo 19. Ubuntu installed. It is unlikely that you will get the space available warning, which appeared only because this “install” was to a simulated disk with only 6 gigabytes.]



[Photo 20. After double-clicking on the document named “INITIAL_SETUP.txt”, drag its window away from the column of icons at left, and follow the instructions on disk.]



[Photo 21. After mouse-dragging the title bar of the INITIAL_SETUP.txt window.]

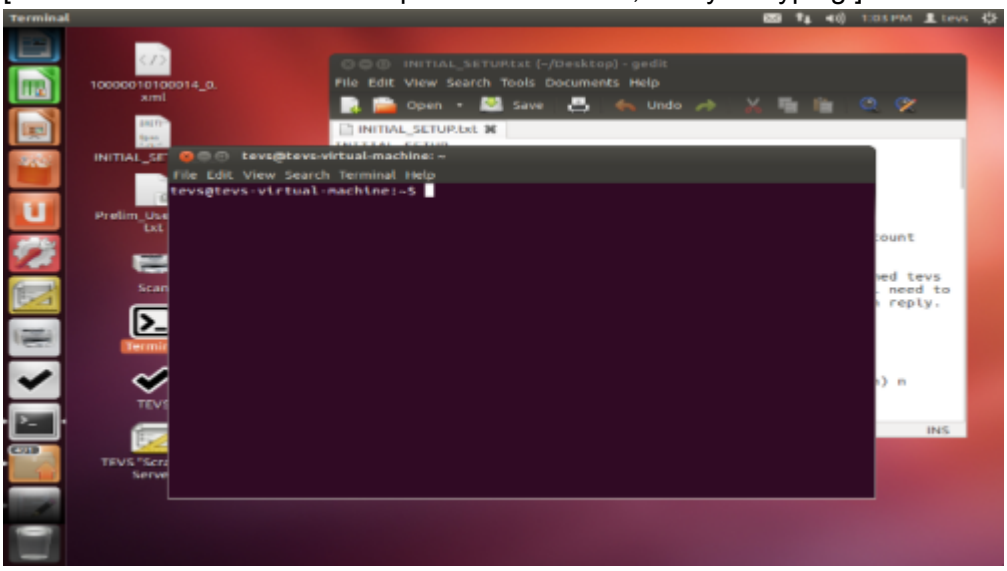


8. Running TEVS

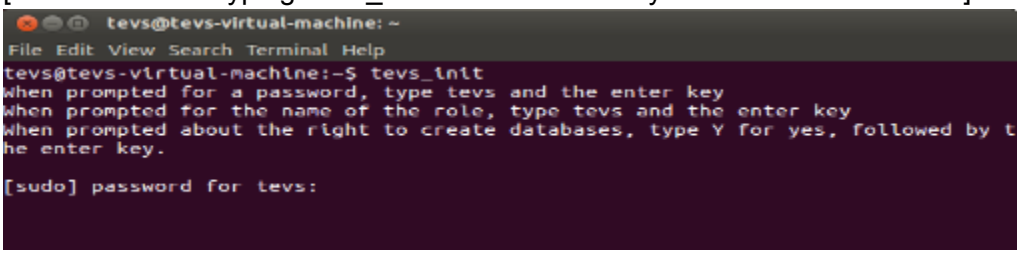
Although we are the only user on our system, Ubuntu Linux supports multiple sessions with multiple users. Communications between a user and the system generally takes place via what is called a “terminal window,” which acts like an old-fashioned computer terminal attached to a large multi-user computer. The INITIAL_SETUP.txt file instructs us to open a terminal window and type the command “tevs_init” followed by the Enter key. This takes care of some initial housekeeping details for the version of TEVS on the disk.

After double-clicking on the picture of a terminal (“>_” inside of a rectangle) we will get a terminal window with which we can issue text commands. The line ending with a “$” is called a prompt, and it indicates that the computer is ready for an instruction. The windows on screen can be moved by dragging their title bars with the mouse button pressed. Only one window at a time can “see” our keystrokes; that will be the window whose title bar is highlighted. To select a window to receive keystrokes, click the mouse on its title bar.

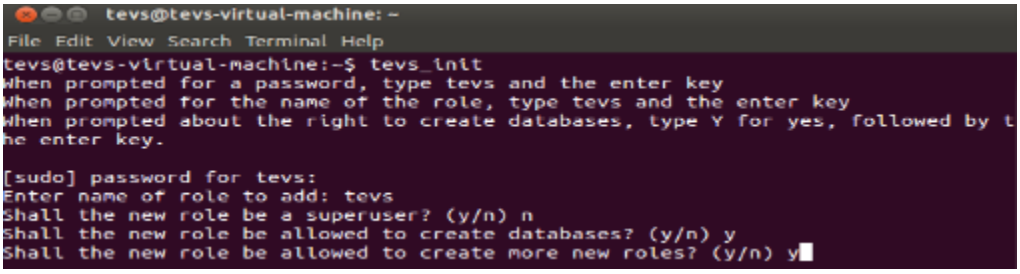
[Photo 22. A terminal window open on our screen, ready for typing.]



[Photo 23. After typing “tevs_init” and the Enter key in the terminal window.]



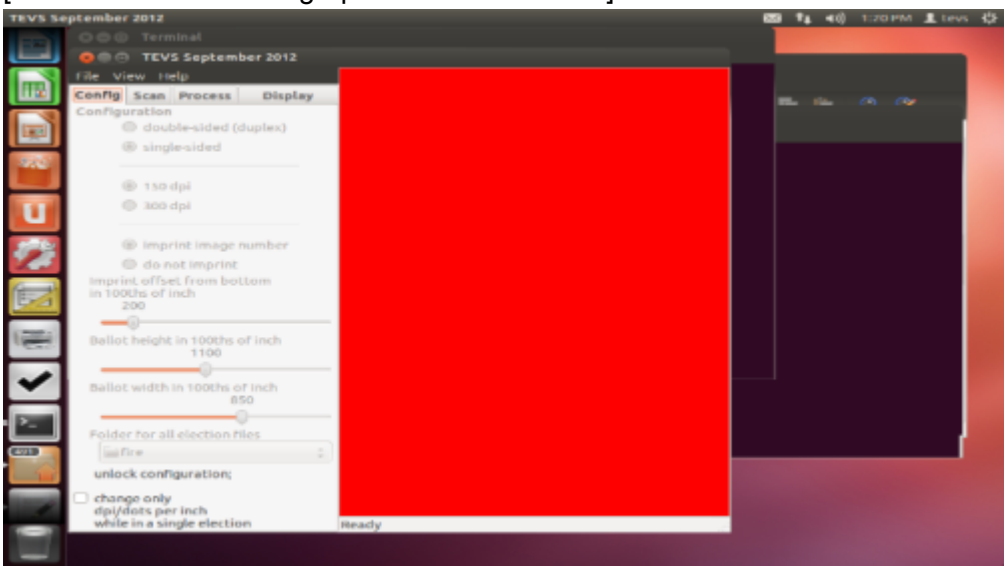
[Photo 24. After typing our password and the Enter key in the terminal window, giving “tevs” as our role’s name, and specifying that, yes, the new role should be able to create databases.]



A new database is set up for TEVS’ use, and various files are unpacked into a new folder named “fire,” after the small Fire District election whose ballots are provided on the DVD. We see the list of files scroll by in the terminal window, followed by a new prompt, again ending with a “$” sign. That means the terminal is ready for us to provide additional instructions.

Rather than continue typing instructions into the terminal, we will use the graphical user interface, which is just another way of signalling our instructions to the computer. First, we’ll start the TEVS program itself by double-clicking on the “checkmark” labelled “TEVS.”

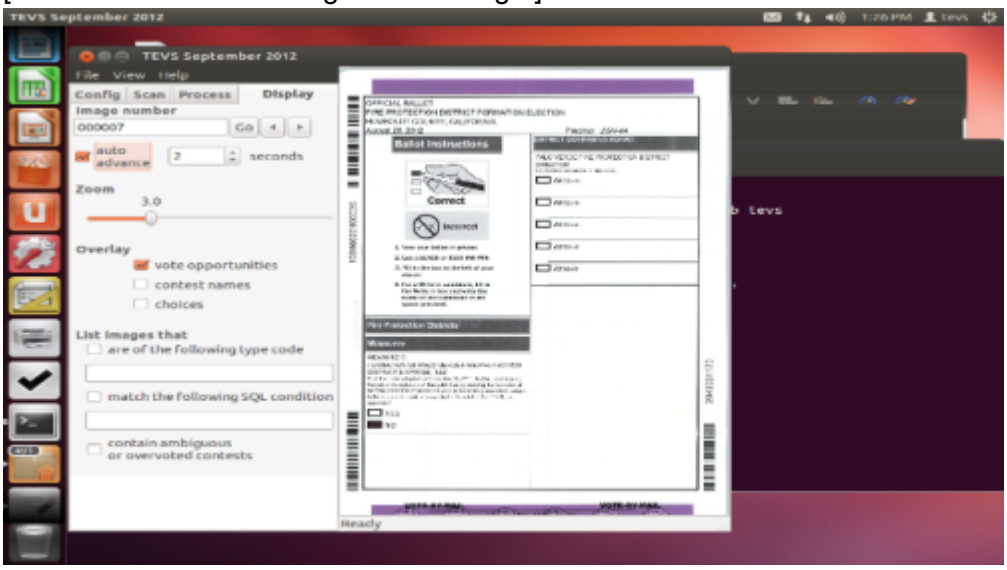
[Photo 25. The TEVS graphical user interface.]



Important warning: This version of TEVS can control scanning only for the fi-5900C scanner. Only minor changes are required to control different scanners. TEVS uses the open source SANE protocol to communicate with scanners, but not all scanners will support the imprinting settings we've used with the fi-5900C.

In the TEVS window, click on the fourth tab, labelled "Display." Then click in the box under the words "Image number" and type 1 followed by pressing the Enter key or clicking in the "Go" button. An image of the first ballot file will appear. As TEVS has not yet built a marks database, no indication of votes appears. To run through the ballot images, click the up-arrow next to the word "seconds" to specify a pause of 2 seconds, then check the box to the left of the words "auto advance." The ballot images will change approximately every two seconds.

[Photo 26. TEVS showing a ballot image.]

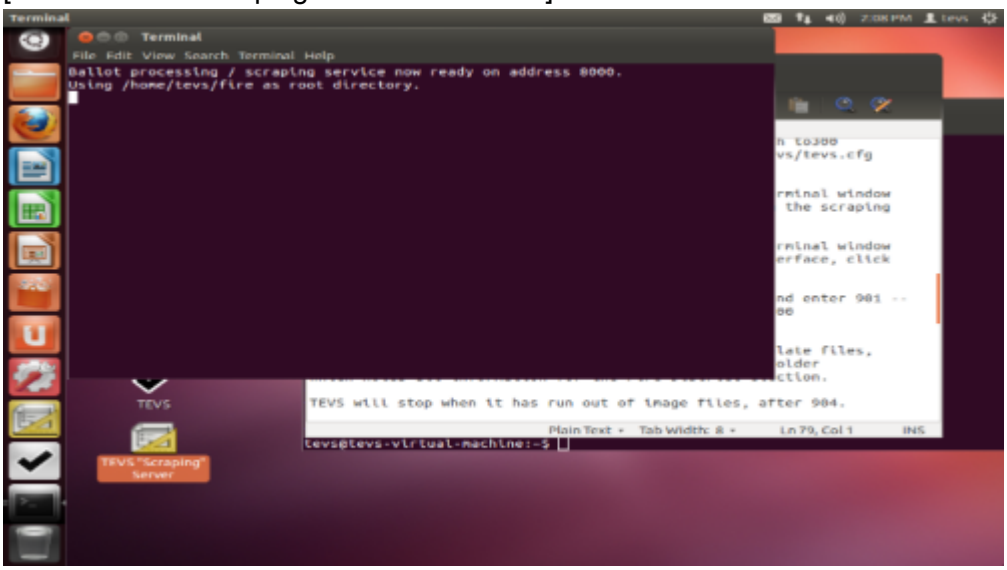


Exit the TEVS program by selecting File->Exit from the menu, or by clicking the “X” in the upper left corner of its window, left of the words “TEVS September 2012.”

Aspects of TEVS behavior will be determined by the contents of a text configuration file named tevs.cfg. The demonstration disk includes two different versions, one set for 150 dpi images and the other for 300 dpi images. In order to have TEVS determine the contests and choices directly from initial ballot images, we will give it higher resolution images of each ballot style at 300 dpi, and so must set the configuration file to accommodate this resolution. To do so, open a terminal window or go back to an open terminal window, and type, without the quotes “./to300” -- dot, slash, “t” “o” “3” “0” “0” -- followed by the enter key. This copies an appropriate version of the configuration file to the name tevs.cfg.

Only after you have the correct configuration file by typing ./to300, start the program which extracts information from ballots, called here the “TEVS ‘Scraping’ Server,” by double-clicking on the image at the bottom of the column of pictures. The result will be a terminal window used by the scraping server to output debugging messages. It will look like this:

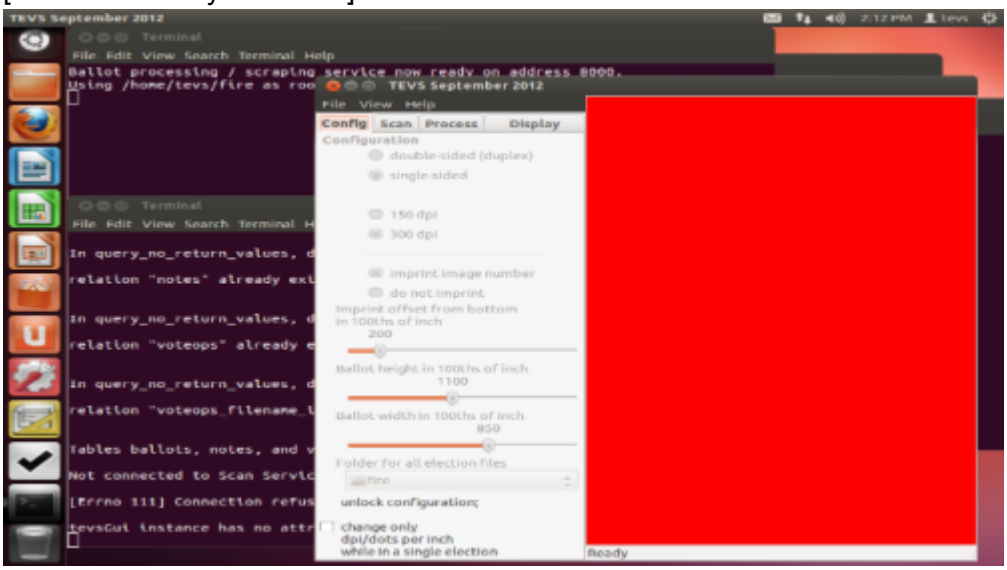
[Photo 27. The scraping server's Terminal.]



Notice there is no "$" sign. This terminal window is not prompting you for information.

Now double-click on the checkmark icon representing the TEVS graphical interface. You'll notice that the TEVS graphical interface also has an associated terminal window for debugging output. You can rearrange windows on your screen like this:

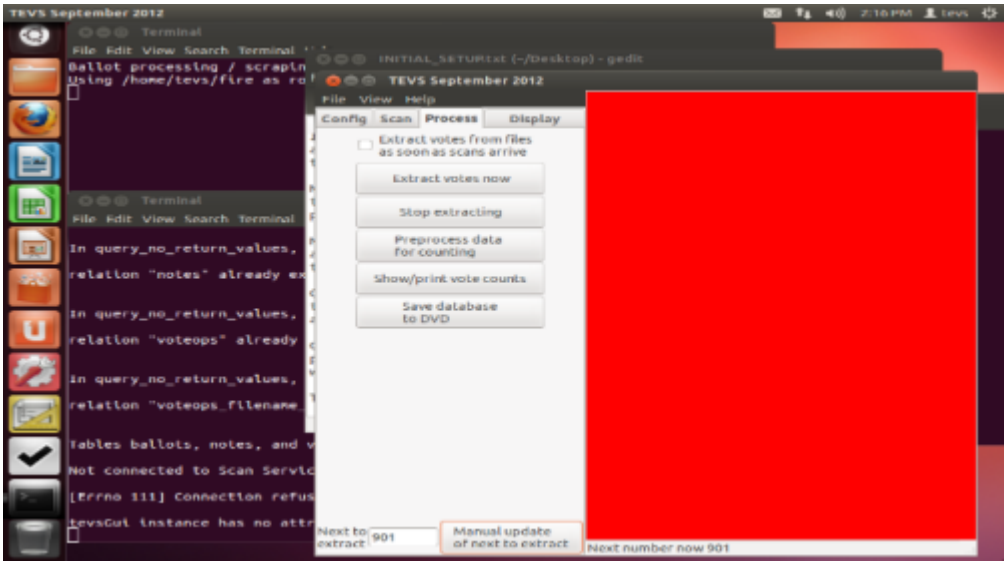
[Photo 28. Many windows.]



The DVD contains images from a small Humboldt County Fire District election as well as 300 dpi images of one ballot of each ballot type. The 150 dpi ballot images are numbered 1-500. The 300 dpi images of each ballot type are numbered 901 through 904. These are the ones we will process first, as templates for the others. Click on the "Process" tab of the TEVS window, then press the button near the bottom, "Manual update of next to process," and enter 901 in the

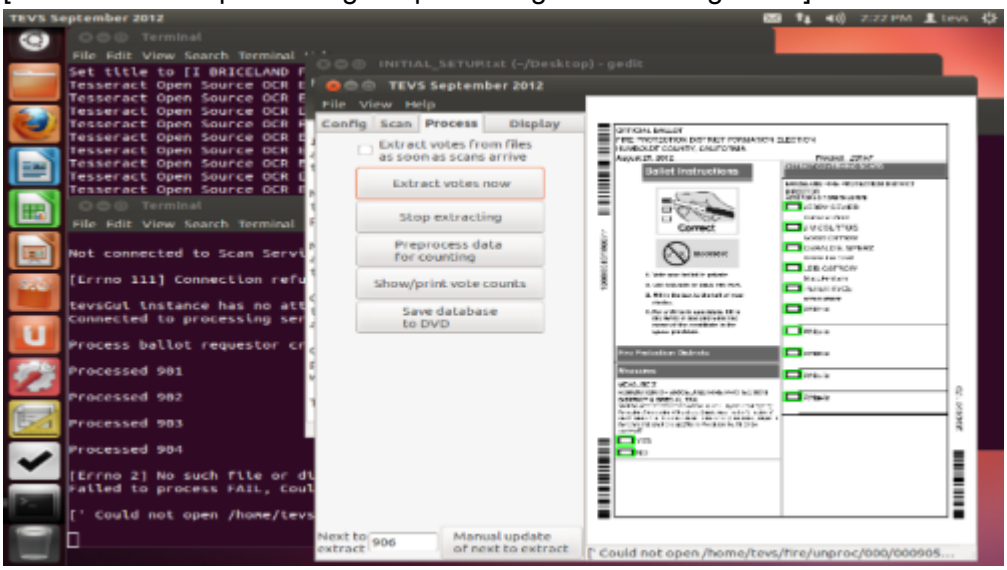
popup dialog box that will appear. Click the OK button to dismiss the dialog box. Your screen will look like this:

[Photo 29. Ready to process templates.]



Press the top button on the “Process” tab, labelled “Extract votes now.” TEVS will read in image 901 and will recognize that it does not yet have a template for ballots of its type. It will call a series of programs to assist it in analyzing the image to divide it into contests and choices. In the debugging output terminals, you will see many calls being made to the program “Tesseract,” which is an open source optical character recognition used to read stretches of text TEVS believes corresponds to contests and choices. As each of the four images 901 through 904 is processed, TEVS finds it has no corresponding template and builds a template. Once it has built a template from image 901, it applies it to the image to search for votes and redisplays the image with green rectangles around each vote target, indicating “processed target and did not find any mark.” The same will happen on images 902, 903, and 904. When TEVS tries to load image 905 it will find no such image, and will report that at the bottom of its window:

[Photo 30. After processing template images 901 through 904.]



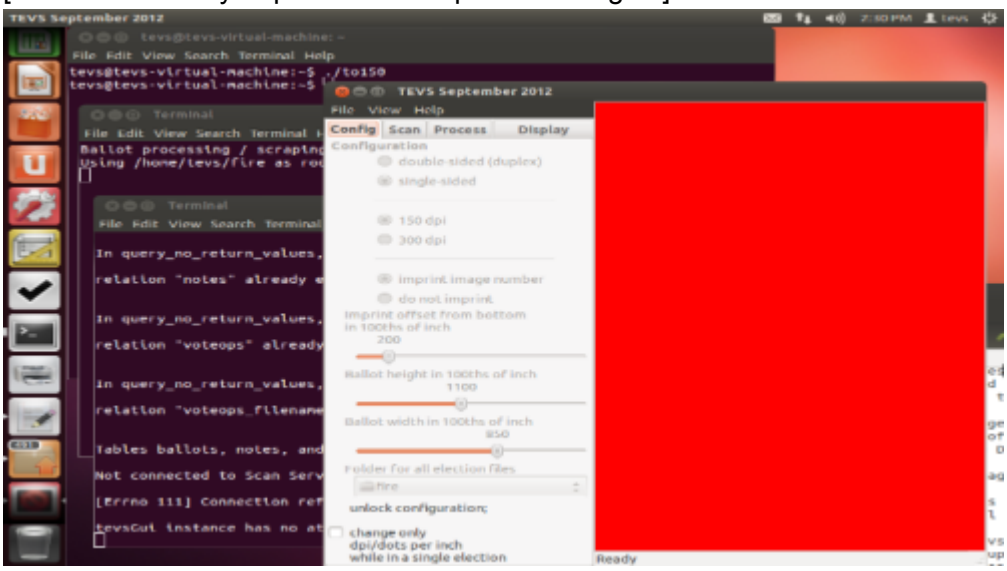
(If you are curious about the files TEVS will have generated and are comfortable navigating the Linux filesystem, they are in /home/tevs/fire/templates.)

Now we are finally ready to process the real ballots, at 150 dpi resolution. To do this we must exit from the scraping program and from TEVS, change the configuration file, and restart the scraping program and TEVS. The most straightforward way to do this is to eliminate all the windows by clicking the circled “X”s in their upper left corners. You can then double-click on the terminal (“>_”) picture to start a new terminal.

In the new terminal, type “./to150” followed by Enter. This copies the appropriate configuration file for 150 dpi images to the name tevs.cfg. Then restart the programs, first the TEVS “Scraping” Server, and then the main TEVS program. Double click on the appropriate pictures.

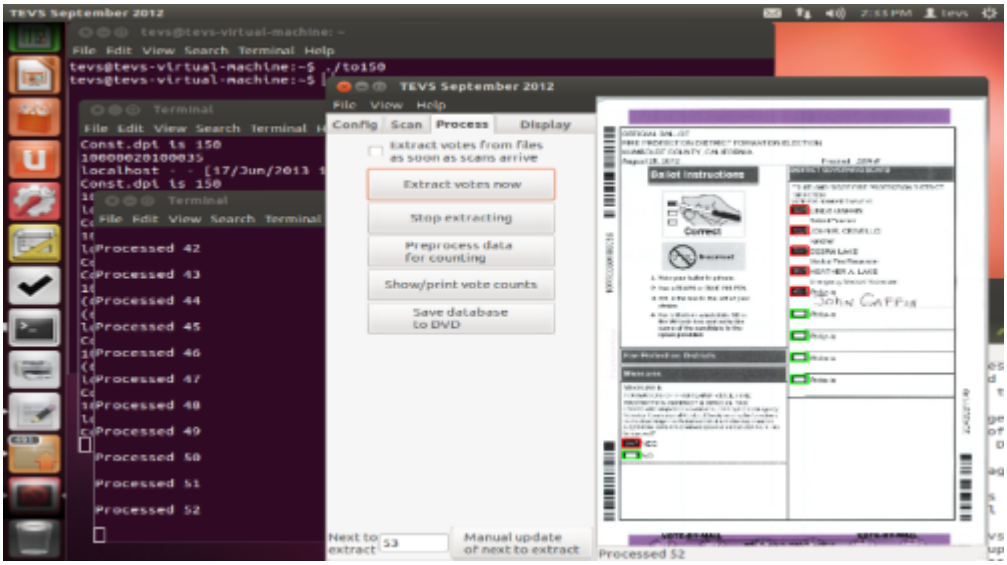
When you’ve started both programs and rearranged your windows, your screen may look like this:

[Photo 31. Ready to process 150 dpi ballot images.]



Our ballots begin at number 1, so we go to the “Processing” tab, click the manual reset button at the bottom, and enter 1 in the resulting popup, then click the OK button of the popup. We can then click the “Extract votes now” button. As each ballot is processed, we will see it displayed with red around the voted rectangles:

[Photo 32. Processing ballots.]



As long as a ballot is of the same type as a template TEVS already knows about, it does not need to perform the slow analysis, but instead uses the existing template information to determine where on the ballot to look for votes.

As TEVS is running through the ballots, it is entering information about the vote target areas into the database that you set up by calling “tevs_init.” This is a set of tables in an open source

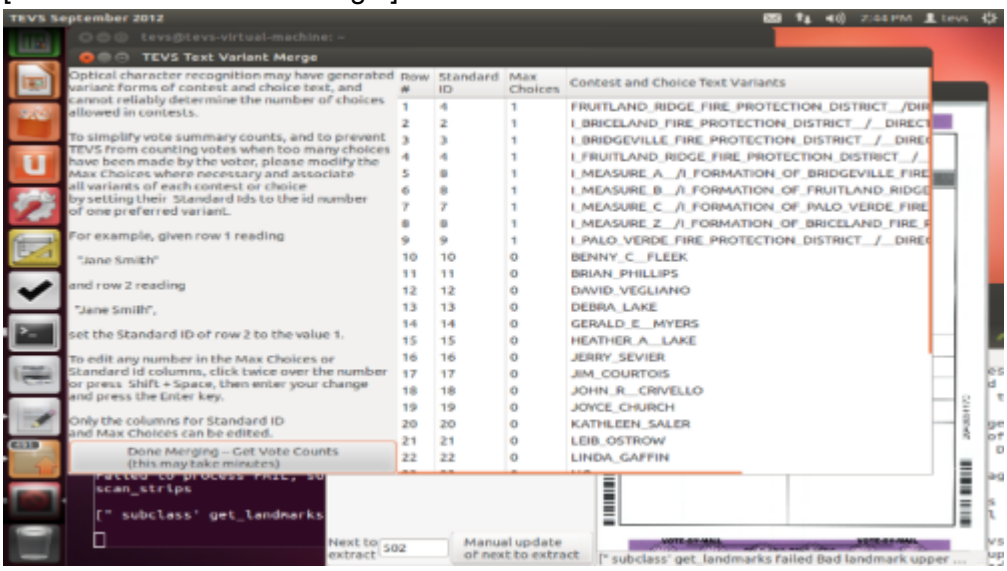
database system called Postgresql.

Ballot processing will stop when TEVS encounters a ballot it cannot process. You can restart processing by advancing the “next to process” number past the problematic image and clicking “Extract Votes Now.” (TEVS should not stop when processing the ballots on the demo DVD.)

Once ballots have all been processed, we need to take care of any failures of the optical character recognition system, and tell TEVS how many votes a single ballot can legally have for each contest. This is necessarily a tedious task, and the software’s usability is still primitive for this task. If you are comfortable with XML, you may wish to go into the template files after they have been created and ensure that each contest and choice has been properly OCR’d prior to scanning the actual ballots. This will reduce the amount of work to be done later.

To perform this task, after processing all 500 ballots, press the “Preprocess data for counting” button on the “Process” tab. A window named “TEVS Text Variant Merge” will appear. You can enlarge it by dragging its upper right corner. It will look like this:

[Photo 33. Text variant merge.]

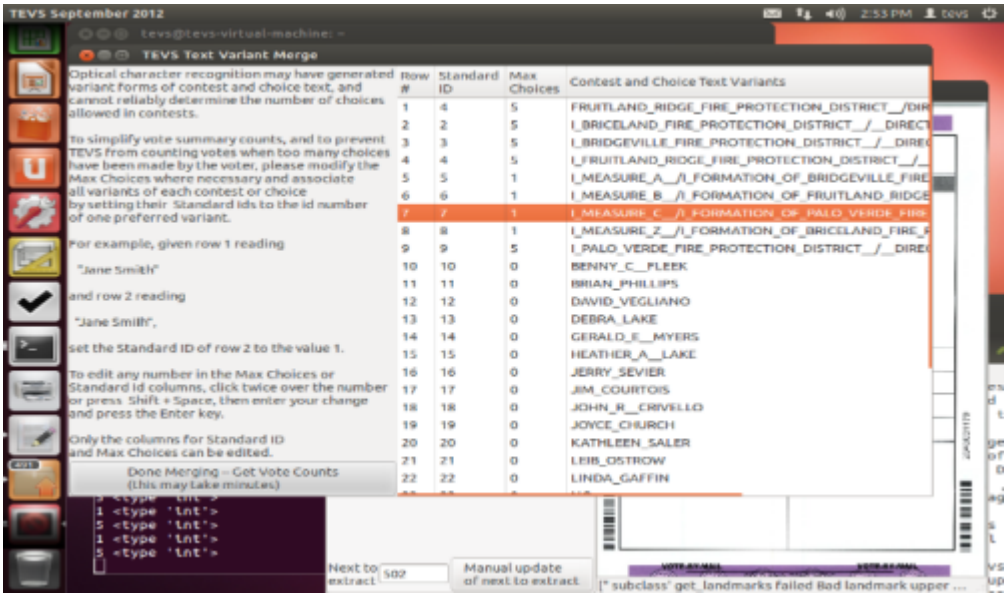


For each contest, enter the maximum number of votes allowed into the corresponding field in the “Max Choices” column. For each district’s directors, this will be 5 votes, while for the various Measures the correct maximum is one vote (yes or no).

Also, we can notice that the Fruitland Ridge Fire Protection District Director election appears twice, first on row 1 and then on row 4, preceded by a false “I” where the OCR misinterpreted the column boundary line. We want these two contests to be recognized as the same, and we do that by “pointing” row number 1 at row number 4 (or row number 4 at row number 1). Because the text of the two variants was very nearly the same, TEVS has already tried to do this, putting a “standard id” of 4 into row 1. That’s fine, but TEVS has guessed wrong on merging Measures A, B, and Z, which it thinks are all the same contest, line 8. We must go into

the Standard ID column and correct rows 5 and 6 to have standard id's of 5 and 6, so that things look like this:

[Photo 34. Corrected merge information.]

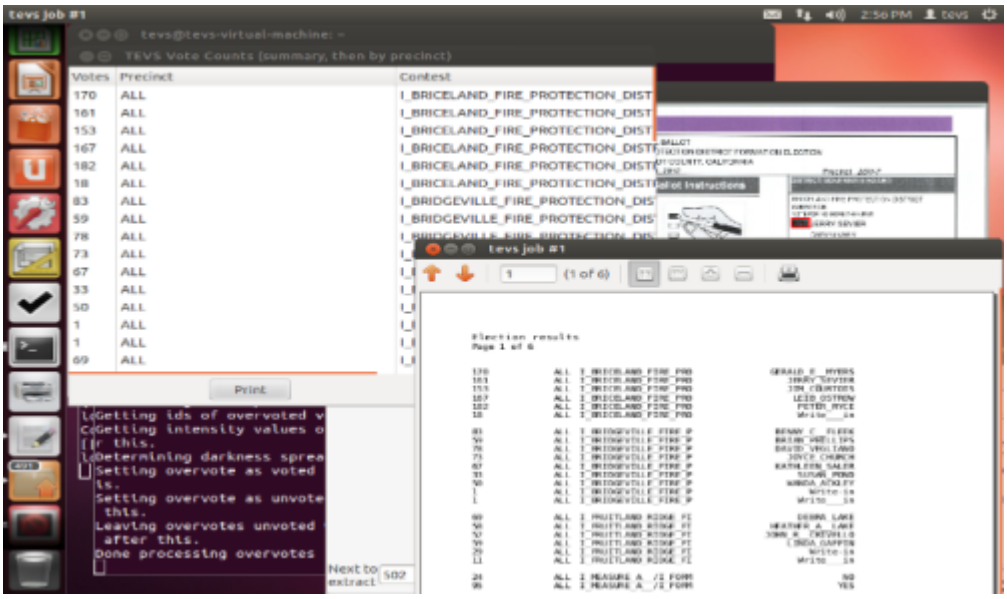


Common OCR misreads may include one for lower case "L," zero for upper case "O," and less obvious replacements such as "IN" for "W."

Warning: If you find many seriously poor misreads, it is possible that TEVS has failed to recognize one of your ballots' codes and has built a template from a 150 dpi image rather than a 300 dpi image. In such cases, it is probably easier to modify the problematic template, replacing its contents with the contents from the template representing the correct code, but leaving it in place for when the incorrect code is read. Once the problematic templates are redone, you would clean the database and process ballots again from scratch.

Only after we have merged all variants and specifying all max votes for contests where more than one vote is allowed can TEVS count what it believes to be valid votes. We press the "Done Merging" button to have TEVS do this. A prompt asks us if we've processed ballots since viewing counts, we reply "Yes." TEVS then presents us with the vote counts, which may be printed. The next screen shot shows the votes as shown on screen and what appears after Print has been pressed and print preview selected. The counts are summarized over all ballot types and, on subsequent pages, subcounts are given for each individual ballot type, which generally corresponds to a precinct. This version of TEVS does not insert the precinct, instead identifying precinct by the template file that was used to generate the template used to analyze a ballot. Template 901 corresponds to one precinct, 902 to another, and so on.

[Photo 35. The counts.]



Prior to comparing TEVS results with official results, you must look through TEVS logs to determine which ballots were not processed. Typical reasons for a ballot not being processed include problems identifying landmarks on the ballot, excessive tilt or skew, and problems interpreting the bar code or identifying pattern successfully.

The number of ballots that are not processed should be substantially less than 1% of the total input batch, but these ballots must be counted manually, and the TEVS vote totals must be adjusted to reflect these unprocessed ballots.

Also, all overvotes must be manually examined. Overvotes are situations where more than the allowed number of targets have been marked in a contest. Overvotes may really be a mistake on the part of the voter, or they may have corrections that TEVS does not understand. On Hart ballots, instructions tell the voter to make a cross through an incorrect marked target and draw an arrow with the word “Yes” pointing to the correct marked target. TEVS makes no attempt to detect this, so you must resolve all overvotes. On the “Display” tab of the main TEVS window, under “List images that”, you may check the box next to “contain ambiguous or overvoted contests.” A list of ballot images with overvotes or detected ambiguities will appear, and you can click on each item to view the corresponding ballot image..

Interacting with the generated database

TEVS has now built two tables, one corresponding to ballot images and one corresponding to individual votes. These can be inspected using any software able to connect with Postgresql. The most basic software that does this is the postgresql monitor program, psql. This is not a visual program. Visual programs are available, but will not be described here.

To run psql, we start by getting rid of all the windows now on our screen except for our first terminal window. (You could start a new terminal if you wish while TEVS is still running; we are just eliminating some clutter.) At the "$" prompt, type psql and the Enter key. The prompt will change to "tevs=>" as a way to remind us we are now using the psql monitor and the database tevs.

At the tevs=> prompt we can type SQL queries. Most queries will require SQL join operations between the ballot table and the voteops table. Here is one example, and the results:

```
select
red_mean_intensity as red, choice_text
from voteops join ballots on ballots.ballot_id = voteops.ballot_id
where filename like '%099.jpg'
```

In English, this means "please merge the ballot and voteops tables, and using the merged table, show me the choices from any ballot images whose name ends 099.jpg, along with the red intensity of the vote target. (Darker areas reflecting less light give lower numbers, with 255 representing full brightness and 0 representing none, so white areas will have a high red intensity, while black will have low intensities on all three color channels, red, green, and blue.)

The first two results appear as

```
red  | choice_text
 62  | YES
154  | NO
```

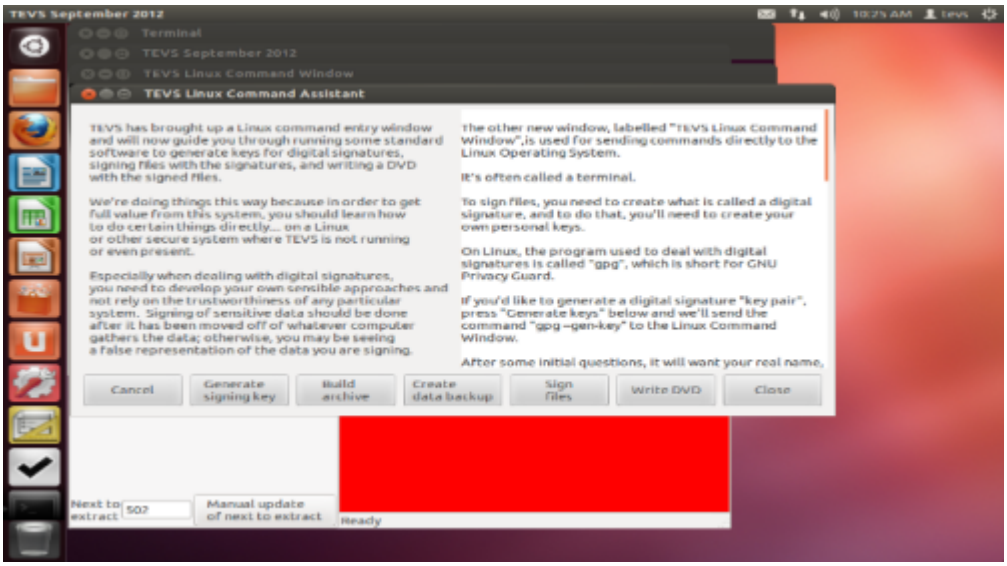
meaning we can infer that a mark was made at the YES target and not at the NO target.

If you are familiar enough with SQL to make sense of the above, please note that the \d command instructs psql to list the available tables, and \d followed by a table name lists the fields of the table. If you are not, you would almost certainly want to familiarize yourself with SQL prior to examining the stored information.

9. Saving and distributing what TEVS has generated

From TEVS main window, select the "Process" tab, then click the button labelled "Save database to DVD." This will open two additional windows, a command entry terminal window controlled by TEVS, and a window intended to guide you through the process of saving and signing your work.

[Photo 36: TEVS Linux Command Assistant Window]



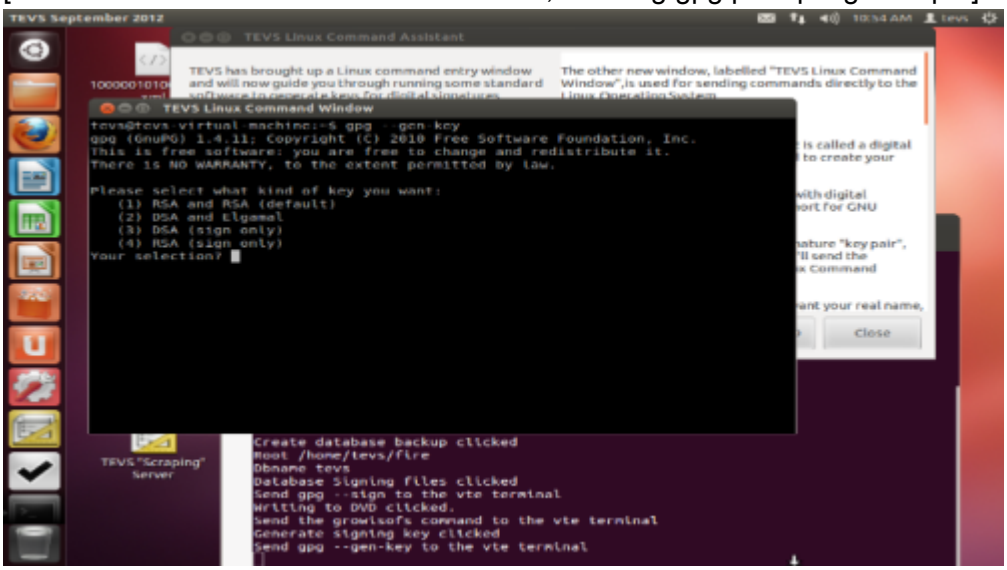
As the instructions in this window suggest, you really ought to involve someone in your process who is familiar with working with digital signatures (and the Linux operating system in general), who can assist you in appropriately copying material to a different machine and then signing it using your own signature and software. However, by clicking the buttons at the bottom of the window, you can use TEVS to help guide you through the basics of the process.

When you click “Generate signing key,” TEVS will enter the necessary command into the Terminal window labelled “TEVS Linux Command Window. You must then click in that window to bring it to the top of the window stack, and answer the questions posed. I’d suggest following the defaults except for expiration. Rather than providing no expiration date, have your key expire at a date after which you no longer need to protect the image’s integrity with your signature. This could be, for example, a month after the window for election challenges has passed.

As part of the dialog with the “gpg” digital signature program, you will actually be called upon to move the mouse around to generate sufficient randomness for the key to be created. This is normal.

When this process has been completed, return to the window labelled “TEVS Linux Command Assistant” and click the buttons labeled “Build archive,” “Create data backup,” and “Sign files.” Each step will send a command to the “TEVS Linux Command WIndow.” Finally, click “Write DVD.” TEVS will not actually write the DVD; to write a DVD, you could duplicate the command sent to the command window, removing the “-dry-run” argument.

[Photo 37: TEVS Linux Command Window, showing gpg prompting for input]



Note that the digital signature you have created is completely unsafe if others have access to the computer you are using -- in that case, your private key is protected only by your passphrase, which is insufficient for security. In reality, you should create a signing key on a system that others cannot access, or keep the private key on a removable USB drive that you keep in your possession. The details of this are beyond this document.

10. Further usage information

Configuration

Aspects of TEVS behavior will be determined by a text configuration file named tevs.cfg. Depending on how you've received TEVS, you may wish to alter these settings. The main settings you may wish to change are the "root" folder, the "resolution," the "vendor," the "dark pixel threshold" and the "intensityThreshold."

"**root**" (within the "Paths" section) points to a folder on the system in which all templates, logs, and scans will be stored. If you would like the system to store material on your computer's hard disk, you may "mount" that disk and point the root to a filesystem on that disk. This is done using the "mount" command; see any Linux manual for further information. Note that when you have finished your processing, all relevant material except the database itself will be present within the folder yo specified as "root" -- this folder can be copied to a backup device and encrypted and/or signed as protection against unauthorized changes.

By default, the root for the default user "tevs" will be /home/tevs/data.

"**ballot_dpi**" in the "Scanner" section Is the resolution at which TEVS scans ballots; "**template_dpi**" is the resolution at which TEVS scans templates.

“**brand**” in the “Layout” section specifies the overall ballot design. Different vendors have different ways of laying out their ballots, and different ways of encoding information about the layouts. TEVS is still being developed, on a hobby basis. Most work is being done to enable Humboldt County’s Election Transparency Project to process our Hart brand ballots. However, the source code is structured to allow the development of modules handing different brands. You will find modules called “Basichart” and “Basicdiebold” that will work somewhat. Unfortunately, the effort to maintain versions of TEVS for vendors other than Hart is beyond my abilities or inclinations.

If your layout is unsupported, you will need to hire a programmer to extend TEVS so that it understands your layout, or you may wish to develop an equivalent program that performs the same tasks TEVS performs but in a cleaner and more well-documented way. The source code as it exists is on the DVD, in the folders /usr/local/lib/python2.7/dist-packages/tevs and /usr/local/lib/python2.7/dist-packages/scan. Unfortunately, the source code is extremely messy and disorganized. Over time, I hope to continue cleaning it up. In the meantime, the files Ballot.py, BallotSide.py, basichart_ballot.py and basichart_ballot_side.py are your best starting places. Note that the actual ballot analysis for Hart ballots is currently performed by a C program using the leptonica image processing library, in the directory /home/tevs/leptonica-1.68/prog/ballot.c

“**dark_pixel_threshold**” and “**vote_intensity_threshold**” (“Votes” section)

TEVS will examine particular regions of the ballot to determine if they have been marked. The software builds a record (a database) containing information about how bright or dark each region was, and how many pixels in the region were darkened to particular levels. Based on that information, it decides whether the region contains a vote. “dark pixel threshold” and “intensity threshold” are used in the rules, as described below. It’s IMPORTANT to understand that the dark pixel threshold will be different depending on the resolution of your images: at 300 dpi, a vote region will contain four times as many pixels as at 150 dpi.

Rules for votes

(1) if the number of pixels falling into the bottom half of the intensity range exceeds “dark pixel threshold,” it’s a vote;

(2) if the average intensity throughout the range is beneath an intensity threshold, it’s a vote;

(3) if there is a disagreement between (1) and (2), it is both a vote and “suspicious”;

(4) if the average intensity is above the “problem_intensity_threshold,” something is wrong, because even an empty vote target should not have an intensity higher than this threshold;

(5) following the initial processing of all ballots, a review pass checks for situations where there

are more “vote” marks than legal votes for a contest on any ballots. Such situations are flagged as “overvotes,” the “vote” status is removed, and “suspicious” status is set.

(6) A human should examine every instance of a mark considered “suspicious” to adjust TEVS generated totals as necessary.

“Suspicious” marks are rare. As TEVS development continues, more rules can be added to reduce the frequency of marks about which TEVS cannot offer a final determination. (For example, many overvotes are in reality a vote for one choice with a light or erased mark for the other choice. Once enough data has been accumulated, it will be possible to allow TEVS to resolve such situations without necessarily setting the “suspicious” flag.)

Development of any program like TEVS will be a constant tradeoff between running speed and depth of analysis; there is no “right” answer to this tradeoff except one you find satisfactory.

File naming conventions

If you are using TEVS to scan ballots, it will generate JPG format ballot scans and store them in folders within the “unproc” folder, itself inside the folder you have specified as “root”. These folders will be named 000, 001, and so on, and will each contain 1,000 images: 000 contains 000000.jpg to 000999.jpg, folder 001 contains 001000.jpg through 001999.jpg, and so on.

If you have used another program to generate scans, you will want to move them into the same folder structure that TEVS would have created: unproc/000 containing 000000.jpg through 000999.jpg, unproc/001 containing 001000.jpg through 001999.jpg, and so on. All scans should be right side up. (Linux includes the graphics magick tool which you can use to flip upside down images -- type “man convert” into a terminal window to see the instructions for this program’s convert utility.)

TEVS is designed to allow you to scan files and process scanned files simultaneously.
As each ballot is processed, it will appear together with TEVS interpretation of votes (prior to any overvote processing). Ballot processing will stop when TEVS encounters a ballot it cannot process. You can restart processing by advancing the “next to process” number and clicking “Process Now.”

Templates

In order to extract votes from any ballot, TEVS must have a self-generated or manually generated template file. These template files will be located in the “templates” folder within the “root” folder. (On the demo DVD, you will find them in /home/tevs/fire/templates after you have processed template images 901 through 904.) If you are using TEVS self-generation feature with 300 dpi scans, you should allow up to two minutes for the generation of each double-sided ballot’s templates.

During the template generation process, TEVS will be writing information to its log files, which are located in the "logs" folder within the root folder. (On the demo DVD, /home/tevs/fire/logs.) The log named processing_log.txt will contain information about the template generation process. The amount of information will vary depending on the setting of the configuration file's "debug" setting: Debug, Info, Warning, Error.

The generated templates are in a format called XML (somewhat similar to HTML). This format may be viewed and edited in any text editor or in code editing modes of many word processors. The Linux operating system includes "gedit," a program suitable for editing these files.

One template file is generated for each side of each ballot style. The particulars of the ballot style will begin with <BallotSide> and end with </BallotSide>. Regardless of the vendor, the Ballot Side will begin with x and y values for four landmarks, starting with the <Landmarks ...> tag and ending with the </Landmarks> tag. The landmarks are identified as upper left corner (ulc-x, ulc-y), upper right corner (urc), lower right corner (lrc) and lower left corner (llc). "-x" values refer to the "x" coordinate of the landmark, its offset from the horizontal edge of the image; "-y" locations refer to its "y" coordinate, its offset from the top edge of the image.

The landmarks are specified in the units identified within the Ballot Side tag, typically inches.

The landmarks will refer to locations on the image used to generate the template. The exact location will vary from vendor to vendor, but will typically be locations at the extreme outward corner of marks located near the limit of the ballot. If an image is skewed, landmarks may be at negative x locations.

Enclosed within these tags will be boxes, beginning with <Box> and ending with </Box>. Within boxes, the individual choices are delimited by <Vote> and </Vote>.

Within the <Box …> tag are values specifying the contest text ("text"), and the offsets into the image of the box's upper left and lower right corners: x1 and y1 represent the x and y coordinates of the upper left corner, x2, and y2 represent the coordinates of the lower right corner.

Within the <Vote> tag are values specifying the choice text ("text") and the offsets into the **box** of the target area in which a vote would be registered. As with the box, x1 and y1 represent the x and y coordinates of the vote target area's upper left corner, and x2 and y2 represent the lower left corner's x and y coordinates.

When the units value for the Ballot Side is inches, an enclosed box's x1 value is "1.0" and a vote within the box has an x1 value of "0.5", the vote target area for the specified vote is at 1.5 inches from the left edge of the image.

Once a template has been self-generated, you may edit any of the values that have been created. You may also manually create a template, using the format of any auto-generated

template as a starting point. A sample starting point follows, taken from one of the templates built on the demo DVD:

```
<!DOCTYPE BallotSide [
          <!ELEMENT Landmarks (#PCDATA) >
          <!ELEMENT Vote (#PCDATA)>
          <!ELEMENT Box (Vote*)>
          <!ELEMENT BallotSide (Landmarks, Box*)>
          <!ATTLIST BallotSide layout  CDATA #IMPLIED
                    precinct  CDATA #IMPLIED
                    units  CDATA #IMPLIED
                    target-height  CDATA #IMPLIED
                    target-width  CDATA #IMPLIED
                    src CDATA #IMPLIED>
          <!ATTLIST Landmarks ulc-x  CDATA #IMPLIED
                    ulc-y  CDATA #IMPLIED
                    urc-x  CDATA #IMPLIED
                    urc-y  CDATA #IMPLIED
                    lrc-x  CDATA #IMPLIED
                    lrc-y  CDATA #IMPLIED
                    llc-x  CDATA #IMPLIED
                    llc-y  CDATA #IMPLIED>
          <!ATTLIST Box x1 CDATA #REQUIRED
                    y1 CDATA #REQUIRED
                    x2 CDATA #IMPLIED
                    y2 CDATA #IMPLIED
                    max-votes CDATA #IMPLIED
                    text-code CDATA #IMPLIED
                    text CDATA #REQUIRED>
          <!ATTLIST Vote orient CDATA #REQUIRED
                    x1 CDATA #REQUIRED
                    y1 CDATA #REQUIRED
                    width CDATA #IMPLIED
                    height CDATA #IMPLIED
                    text-code CDATA #IMPLIED
                    text CDATA #REQUIRED>
          ]
>

<BallotSide
 layout-id='10000010100014'
 src='/media/other/fire/unproc/000/000901.jpg'
 units='inches'
 party='ENDOFPARTY'
 precinct='_media_other_fire_unproc_000_000901_jpg'
```

```
 target-height='0.1700'
 target-width='0.3300'>
 <Landmarks
   ulc-x='0.5800' ulc-y = '0.6667'
   urc-x='7.9000' urc-y = '0.6667'
   lrc-x='7.8967' lrc-y = '10.3167'
   llc-x='0.5767' llc-y = '10.3100'
 />
 <Box x1='0.5867' y1='6.9267' x2='4.2367' y2='8.9133'

text='I_MEASURE_A__/I_FORMATION_OF_BRIDGEVILLE_FIRE_PROTECTION__/I_DIS
TRICT___SPECIAL_TAX__/I_Shall_the_order_adopted_on_November_16__2011__
by_the_Local_Agency__/I_Fo'
  text-code='0'
  max-votes='1'>
  <Vote orient = 'V' x1='0.0967' y1='1.4200' x2='0.4300' y2='1.5967'
   text='YES'
   text-code='YES'>
  </Vote>
  <Vote orient = 'V' x1='0.0967' y1='1.7100' x2='0.4300' y2='1.8867'
   text='NO'
   text-code='NO'>
  </Vote>
 </Box>
 <Box x1='4.2367' y1='1.9967' x2='7.8667' y2='8.8000'

text='I_BRIDGEVILLE_FIRE_PROTECTION_DISTRICT__/__DIRECTOR__/I_VOTE_FOR
_NO_MORE_THAN_FIVE__'
  text-code='0'
  max-votes='5'>
  <Vote orient = 'V' x1='0.1000' y1='0.6133' x2='0.4333' y2='0.7900'
   text='KATHLEEN_SALER'
   text-code='KATHLEEN_SALER'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='1.1133' x2='0.4333' y2='1.2900'
   text='DAVID_VEGLIANO'
   text-code='DAVID_VEGLIANO'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='1.6133' x2='0.4333' y2='1.7900'
   text='BENNY_C__FLEEK'
   text-code='BENNY_C__FLEEK'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='2.1133' x2='0.4333' y2='2.2900'
   text='JOYCE_CHURCH'
   text-code='JOYCE_CHURCH'>
```

```
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='2.6133' x2='0.4333' y2='2.7900'
   text='WANDA_ACKLEY'
   text-code='WANDA_ACKLEY'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='3.1133' x2='0.4333' y2='3.2900'
   text='SUSAN_POND'
   text-code='SUSAN_POND'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='3.6133' x2='0.4333' y2='3.7900'
   text='BRIAN_PHILLIPS'
   text-code='BRIAN_PHILLIPS'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='4.1133' x2='0.4333' y2='4.2900'
   text='Write-in'
   text-code='Write-in'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='4.6533' x2='0.4333' y2='4.8300'
   text='Write___in'
   text-code='Write___in'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='5.1967' x2='0.4333' y2='5.3733'
   text='Write-in'
   text-code='Write-in'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='5.7400' x2='0.4333' y2='5.9167'
   text='Write-in'
   text-code='Write-in'>
  </Vote>
  <Vote orient = 'V' x1='0.1000' y1='6.2767' x2='0.4333' y2='6.4567'
   text='Write___in'
   text-code='Write___in'>
  </Vote>
 </Box>
</BallotSide>
```

The template format is unforgiving: you must close every open angle bracket with a close angle bracket, you must close every open single quote with a closing single quote, and you must identify settings exactly as they appear: ‘layout-id’ is not the same as ‘layout id’.
The layout-id field of the ballot side should be the same as the layout id portion of the file name used to hold the template. For example, file 1234_0.xml represents the 0th side of the ballot whose layout id is 1234; the BallotSide’s layout id value within the file should be ‘1234’.

Each vendor uses a different technique for identifying different layout styles. It’s important to understand that multiple layout styles may have the same visual appearance, with the exception

of the layout style code and, perhaps, an identifier somewhere on the ballot.

For example, a jurisdiction may have several precincts which share an identical set of candidates and races, but each use ballots printed with a different precinct id. It is likely that each precinct has a different layout style, even though their actual layouts appear identical. The layout coding is also being used to identify the precinct to the tabulation system.

TEVS does not examine each ballot for its layout, relying instead on the coding technique used by the ballot vendor. This is because examining each ballot for its actual layout would be far more time consuming than examining each ballot for the layout style encoding, which typically occupies only a small portion of the ballot. Every ballot's layout style encoding is determined by examining the appropriate region, and if the encoding has already been encountered, TEVS retrieves the pre-built layout.

Warning: TEVS can easily be fooled by modified ballots whose layout code does not correspond to its actual layout.

The vendor encodings that TEVS currently uses are:

Basichart: bar code in the upper left corner
Basicess: pattern of boxes down the left hand side
Basicdiebold: pattern of dashes across the bottom

[Appendix 1: Generating images suitable for template creation from Hart printer PDFs]

In the past, we have used printer PDF files to generate 300 dpi template images, rather than using scans of actual ballots. The process we have used to go from printer PDF files to 300 dpi JPGs is as follows:

1) split the printer PDFs and extract only the pages representing the first ballot of each type
2) convert these pages to 300 dpi JPG files
3) crop these pages at the location where the physical ballots are perforated
4) renumber the pages to fit TEVS numbering expectations

1. Splitting the printer PDFs

The printer PDF files contain one page for every ballot side to be printed. Using the open source program pdftk, we extract only the first page or, if the ballot is double-sided, the first two pages.

The command for each file is pdftk input.pdf cat 1-2 output outfile.pdf; to repeat this process for all files in a directory, the bash shell command would look like this:

for x in *pdf; do echo $x; pdftk $x cat 1-2 $x; done;

2. Converting the PDFs to JPGs

To convert the single page PDF files to 300 dpi JPGs, we use the open source suite Image Magick. The command “convert” allows us to convert the PDFs to JPGs. We use the geometry argument to specify that the image should be 8.5 x 18 inches and rendered at 300 dots per inch, or 2550 dots wide by 5400 dots tall.

convert input.pdf -geometry 2550x5400 output.jpg

To convert each pdf file in a directory to a jpg with the same base name, the bash shell script would be:

for x in *.pdf; do echo $x; convert $x.pdf -geometry 2550x5400 $x.jpg; done;

3. Cropping off the “perforated ballot stub”.

The bottom inch of each jpg file will not be present in the completed ballots; we need to crop this

inch away. Once again, we use the “convert” command from ImageMagick:

```
for x in *.jpg; do
 echo $x;
 convert -crop 2550x5100 ${x}.jpg ${x}crop.jpg;
 rm ${x}.jpg;
 mv ${x}crop.jpg ${x}.jpg;
done;
```

4. We renumber these files to six digit serial numbers, starting from 0.

```
num=0
for x in *.jpg; do
  mv $x ${num}.jpg;
  num=$((${num}+1));
done;
```

5. Finally, we move these files to a temporary directory root for TEVS, template_files

```
mkdir ~/template_files;
mkdir ~/template_files/unproc
for x in {000..099}; do mkdir ~/template_files/unproc/${x}; done;
mv ?.jpg ~/template_files/unproc/000
mv ??.jpg ~/template_files/unproc/000
mv ???.jpg ~/template_files/unproc/000
mv 1???.jpg ~/template_files/unproc/001
mv 2???.jpg ~/template_files/unproc/002 and so on...
```

We can now set TEVS “root” directory in tevs.cfg to /home/tevs/template_files and TEVS will treat the images in /home/tevs/template_files/unproc/ as ballot files, generating templates from them when no template exists. Since there are initially no templates, each file will cause TEVS to generate a template in /home/tevs/templates. These files can then be copied to the regular TEVS root’s templates directory.

[Appendix 2: How does the template generation work?]

The template generation uses the open source Leptonica image processing library to locate vertical and horizontal lines in the image, which are treated as column and contest dividers. Boxes of a suitable size are considered to be vote targets. The contest boxes are divided into lines of text, with the lines above all vote targets treated as contest names and the lines aligned with vote targets treated as choice text. Individual lines are then cropped and passed to the open source tesseract program. The code for template generation is in ballot.c in the Leptonica directory. Earlier template generation attempts in Python were both slower and less robust than the Leptonica approach has proved for Hart ballots. It is not certain how well this approach will work with Diebold and ESS ballot designs, but the basic idea should work effectively with any ballot consisting of contests enclosed in boxes.

[Appendix 3: Reinitialization]

TEVS is really not designed to be reused; instead, it is intended that it be reinstalled from scratch prior to each use.

If you wish to empty out the database to do a replacement run, run psql in a terminal window and issue the following commands in psql:

delete from voteops;
delete from ballots;

This will remove old votes and ballots.

If you wish to rerun the existing images, you will need to move them from the folders under “proc” to the folders under “unproc.”

[Appendix 4: Checking your results]

There are a wide variety of “sanity checks” you may wish to perform on your data. For example, you may wish to make sure that the x offsets of your choices don’t stray too far from the x-offsets of the voting columns. But the best check that the results match the scans is to select the “Display” tab and step through some or all of the ballot images. Rather than manually click to advance from image to image, you can set TEVS to advance on its own every second, every two seconds, and so on. You should be in complete agreement with TEVS results, unless there is additional writing on the ballot -- if you are not in agreement with TEVS results, it’s important to learn why before proceeding. You may need to adjust templates; you may need to adjust thresholds, or you may have found a problem with TEVS.

Even where TEVS interpretation of the vote areas is correct, however, there will still be ballots where a voter has, for example, filled in the wrong bubble and then filled in a second bubble and drawn an arrow to it, or drawn a big X through the incorrect bubble. These will be relatively rare and are likely to fall roughly evenly amongst the choices, but if you want exact numbers, you must manually observe the ballots to search for them.

TEVS generates composite images of each layout code in the folder “composite images.” By default (in the current program) it does not attempt to accurately line up one image with the next, it simply overlays them as they come. Nonetheless, inspecting composites for a particular layout code will reveal any scribbling on any ballots for that layout code. If there are no marks other than in the vote target areas, you don’t need to manually check any ballots of that layout code for scribbles. If scribbles are evident in a composite, unfortunately, you will need to check individual ballots with that layout code to determine which ballot has the scribble.

[Appendix 5: Making your material available]

There are several ways in which you may choose to make your scans and the associated information available.

First, you can provide copies of the DVD you have generated. You can provide these by mail or in person. You can also provide a copy of TEVS to anyone, so that they can view the data themselves.

Another alternative is to distribute the data from a web site which serves up individual images on request. If you do this, you will need to provide some way for people to confirm that individual images they look at have not been altered. (The digital signature was applied to a single file containing a collection of images.)

This may not be a concern of yours, or of many of the people interested in viewing the files. Still, be aware that there are many possibilities for individual images to be changed once they have been extracted from the signed collection. You might change the images, of course, or someone with uploading permissions at the web site might change the images. But there are more possibilities than that, ranging from the simple (someone gets a password providing them with upload permission at your web site) to the sophisticated (someone implements an invisible "man-in-the-middle" attack, intercepting requests to your web site and serving responses from their own machine, or modifying responses from your site).

Implementing a reliable way to have confidence in single images transmitted over the internet might involve additional software. It is beyond the scope of both TEVS and the ETP.

[Appendix 6: Extending TEVS]

Extending TEVS to new scanners

(Note: as of now, ETP scanning is done by a separate, standalone program; this program also has the ability off the file menu to zip the scanned files, allow you to sign them, and write the zipped files and signature to a DVD. To sign, you MUST have external media inserted which has a toplevel directory named GNUPG and includes the necessary files that would ordinarily be located in ~/.gnupg.

There is really no reason whatsoever to have TEVS perform the scanning. That it does is simply a result of the Humboldt ETP's concern that proprietary scanning software might alter images. The likelihood that unaltered general-purpose proprietary scanning software would alter images to switch votes seems low, but if that is a concern there are other open source scanning programs available in Linux and Ubuntu.

Scanning in TEVS is done by a python script, tevsgui_xmlrpc_scanning_service.py. This script receives requests from the main TEVS program via the xmlrpc protocol and returns acknowledgements to the main TEVS program.

The scanning script sends setup and scanning requests to scanners using the SANE protocol. As written, the scanning script handles requests for update of the following options:

- resolution (how many dots per linear inch should the scan contain)
- source (from ADF Front, ADF Back, and ADF Duplex)
- imprinter (set to True or False, depending on whether you wish to imprint numbers)
- imprinter value (the number to be imprinted)
- imprinter y offset (the vertical offset of the imprint from the top of the page)

To add handling of additional options will require modification of the scanning script. This should not be difficult for a programmer with some experience in the Python programming language.

To add additional options at the user interface will require modification of the main tevsgui.py script and the tevsgui.glade layout file.

These modifications should not be difficult, as they will mostly involve cutting-and-pasting code for adjusting existing options, modifying them to reflect the desired options.

The only reason scanning is done by a separate process is that's the way it is. It was an attempt to enforce some boundaries between tasks.

Extending TEVS analysis routines

Functions to extract or analyze data from the database populated by TEVS can be standard SQL scripts, and need not involve any other TEVS code. The same applies to functions which will alter the data.

If you wish to gather additional data beyond that already gathered by TEVS, you will need to create new tables or modify the existing tables. Good starting point for modifications to gather additional information about vote targets would be the files BallotSideWalker.py and VOP.py.

Extending TEVS to new ballots

If your jurisdiction uses a ballot style not known to TEVS (or if you wish to improve the handling of a known ballot style), you will need to create new sub-classes of the object types BallotClass and BallotSide.

These object types are defined in BallotClass.py and BallotSide.py, and your best starting point is to examine the existing subclasses Basicxxx_ballot.py and Basicxxx_ballot_side.py. There are several primary tasks each ballot subclass must implement:

1. determine the location of four landmarks near the upper left, upper right, lower right, and lower left corners of each ballot image;
2. read encoded information revealing the layout code of each ballot image;
3. provide an automated or manual mechanism for determining the actual layout of a ballot each time a new layout code is encountered

In addition, you may wish to provide mechanisms to extract a precinct identifier and a party identifier from each ballot.

Tasks one and two are straightforward for most ballots. Searching for landmarks may consist of looking for the first black region of a particular size. The main tradeoff is between the speed of your search and the risk of false positives or code misreads. The proper tradeoff will depend heavily on what, in your opinion, you can expect as the quality of input images, in terms of their resolution, their skewing, and their allowable damage.

Task three is difficult, but you may always require the user to generate the layout files manually, or semi-automate the process by prompting the user for lists of contests and their choices, then prompting them for input locations, perhaps by asking for clicks and drags on a presented image of the ballot. As time goes on, generic routines may become more reliable and flexible, but for now, reliably determining the layout must be done differently for most vendor approaches to ballot design.

Tasks one and two take place for every ballot image encountered, and so need to be reasonably fast, where each jurisdiction will define “reasonable.” Task three may be much, much slower, as it takes place only for each new layout encountered. It is reasonable to expect that there will be hundreds of instances of typical ballot layouts, so even if the time for task three is 100 times that of tasks one and two combined, it will still not make up more than half of the time involved in processing the ballots.

For someone ambitious, there is also a potential task 4, determining on-the-fly what strategy should be used for analyzing the ballot image for layout code, contests and choices. Useable ballot designs should have channels or lines dividing columns and should make it clear to human beings where contests and choices are. Layout code regions could, in theory, be detected by searching for repeating dash patterns or bar codes that vary occasionally from ballot to ballot in a large set of ballots. An ambitious programmer could automate the process of determining what approach to use for generating templates from images.

The result of such work would be a program that could be presented with any ballot image and, without further assistance, break it down and extract votes. Just as the layout analysis of each precinct needs to be conducted only once for each precinct encountered, the “brand analysis” would only need to be conducted once for the entire set of ballots presented. In practice, since most jurisdictions will use only one brand, it probably makes more sense to devote efforts to developing an appropriate protocol for analyzing layouts following the given brand’s idiosyncratic approach to layout. This is a much less challenging (and less interesting) task, but the resulting software remains more dependent on vendors continuing to design ballots as they currently do, and more fragile in the face of apparently minor changes in vendor layout approaches, such as changes in line thicknesses, use of halftoning, and so on.

[Appendix 7: Reviewing your process, and closing thoughts]

It is best practice to review problems encountered and successes achieved at the conclusion of each election. This can take the form of an “all-hands” meeting, at which someone is designated to take notes and provide a copy of the notes to all participants. This sort of meeting can also provide a good opportunity for providing at least token gestures to volunteer participants (certificates of appreciation, gift certificates to local restaurants, awards for “best blah”) to reduce the rate at which participants burn out. The ETP approach actually requires a great deal of labor which has little associated glamor, and will not long survive if most volunteers consider themselves unappreciated, or their time investment ill-valued.

If desired improvements are noted, it is important to determine what resources will be required and to assign the “next steps” to a project participant, along with a “report back by” date. As in all work, the fact that an improvement is proposed does not imply that an improvement will be made; designating responsibility and time-frame at least increase the likelihood that someone will attempt the improvement.

You are doing a service to your community by assisting your elections department in ensuring the integrity of your community’s elections. But in doing so you are also taking on a responsibility to enable others to understand what you are doing and what you are not doing. You have a responsibility to do your utmost to resolve discrepancies between the numbers you come up with and the numbers in the official report.

A bit of time spent with ambiguous ballots should make it clear that the official number is not entirely objective. A bit of thought will reveal that no system based on voter intent will always be completely objective. A system might require a particular accomplishment to complete a vote, like the darkening of a certain percentage of the available space in a vote target. Such a system would be objective but would not reflect voter intent at all times.

Any time you involve computers, you are introducing a possibility for public confusion. Surprisingly, the attitude that “if a computer says it, it must be right,” is still quite common. People especially do not understand the ease with which computer-generated images can be tampered with undetectably; they also do not understand the enormous vulnerability of encryption systems to common theft of keys.

Elections officials have a difficult task, one that has been made needlessly harder by their limited resources, the public acceptance of black box voting equipment, and various misunderstandings of the requirements of the voting process. Voting requires that the vote counts be accurate, but that no individual’s vote can be attached to that individual -- even by the

voter herself. Hand counting could work and may well be the appropriate solution -- it is our misfortune that many of our fellow citizens think rapid results are important enough to rule out hand-counting as an option, but show no interest in providing resources that could conceivably make rapid and verifiable results possible by a mechanism understandable by all.

As someone hoping to be useful, you should be prepared to be dismissed by people on all sides, including many election integrity activists and many elections officials. Good luck, and I hope you can find gratification in your important efforts, independent of results.