CSC 493 – SENIOR PROJECTS

Wonder Backup

Report 9 on the work of Weeks 10 and 11

Sean Davis
11/16/10

This document contains information regarding the results of work completed through week 11. At this point, test cases have been included and, with the exception of some functionality, both the command line and graphical user interface programs are functional. Project updates can be found at http://wonderbackup.sourceforge.net.

Table of Contents

Application Development Section

Project Concept Proposal

- **Purpose:** Wonder Backup is an open source, Python-powered, operating system independent backup solution for use in scenarios from end-users to enterprise solutions. Wonder Backup will likely be licensed under the GNU Public License. # Modified 10/18/10
    - **Context:** The context for this program is to fill a need for a free backup solution for any user. Developed in Python, this program will be easily extensible or adjustable for any in-house enterprise solution.
    - **Goals:** This project aims to develop a Python-powered backup solution that can be used in any operating system that can run Python, including modern distributions of Linux, Windows, and Mac OS X.
    - **Audience:** The intended audience is home users, system administrators, and technical service centers. The program will be able to be controlled via a graphical user interface, command line, or through an answer file. # Modified 10/18/10
    - **Functionality:** Given that the program has a number of interface options; it will be functionally usable by anyone. The availability to customize settings using an answer file allows technical users to create precompiled Linux recovery solutions. Lastly, backups can be made to any external source, whether it is an external hard drive, flash media, or a Windows network share. # Fixed Typographical Error 10/18/10
    - **Milieu:** # Alphabetized and Added New Content 10/18/10
        - Amanda Open Source Backup, http://amanda.zmanda.com/
            - This program is for enterprise solutions. It provides backing up to a central server by a server installation and client web interface.
            - Developed in C and Perl.
        - BackupPC, http://backuppc.sourceforge.net/
            - This program is for enterprise solutions. It provides backing up to a central server by separate client and server installations.
            - Limitations include the inability to backup to locally available storage, and no documented support for Mac OS X.
            - Developed in C++.
        - FBackup, http://www.fbackup.com/
            - This very respectable backup program has a lot of power for a great price: free. It supports only Windows, but allows for free usage with any user, home or commercial. It does automatic backups, compression, network backups, removable media backups, just about everything.

        - Limitations exist only in the fact that this software only runs in Windows.
    - **Novelty:** Wonder Backup will be built in Python and will be self-contained. It requires no server installation and does not necessarily need to be run in the native Operating System being backed up.
- **Resources:** # Added New Content 10/26/10
    - Python Programming Language 2.6+
        - http://www.python.org/
    - Samba Windows Interoperability Suite
        - http://www.samba.org/
    - wxWidgets
        - http://www.wxwidgets.org/
    - wxPython
        - http://www.wxpython.org/
- **Challenges:**
    - Python versioning on Unix systems with Python pre-installed.
    - Graphical User Interface # Modified 10/18/10
    - Incremental Backups
    - Support for older operating systems
    - Multiple directory selection in different interfaces
    - Packaging software for use with Windows
- **Measures:**
    - Software properly runs on each tested Operating System.
    - Command-Line interface is functional.
    - Graphical user interface is functional. # Modified 10/18/10
    - Software packages operate as desired.
- **Future Extensions:**
    - Encryption
    - Service, background backups

Inspiration

- **Motivation:** This project is important to me because I have not found an equivalent, Python-based backup solution. Python is my language of choice and I would like there to be an option to have a fully customizable and extensible backup program that is fully open source and freely available to anyone who seeks a better backup program.
- **Profession:** This project will help my professional growth through the creation of a program that has no current alternative. Filling this void in the software universe will begin to publicize my name as a serious programmer. Combined with the experience I hope to earn by approaching this project through a professional business model, this project will certainly further my professional growth.

Vision and Scope

Wonder Backup is a project which seeks to bring a very free and easily customizable backup solution into the software world. Once complete, the software will offer a range of backup options and will be freely usable and modifiable for any individual or organization. Usable for regular backups, or even as a recovery solution, users will delight in the options presented by the software with the ability to use the software with any semi-modern hardware.

Given the time constraints of this semester, the scope of this project needn't be significantly reduced. Within reasonable scope, this project will have a working software base by the end of the semester. All functionality of the software will be available, though perhaps only by a command line interface. The most difficult part of this project will be the single interface for all systems. As previously considered, this may take form as a web interface, but may also result in other possibilities. Guaranteed to be outside of scope will be a Python 3.x version, though this can be expected after release.

Software Requirements Specifications

1. Traverse a directory structure
    - Evaluation Method: List files in deeper subdirectories.
    - Dependencies: None.
    - Priority: High
2. Copy a single file
    - Evaluation Method: Copy a file from one location to another.
    - Dependencies: None.
    - Priority: High
3. Copy multiple files
    - Evaluation Method: Copy multiple files from one location to another.
    - Dependencies:
        - 02. Ability to copy a single file
    - Priority: High
4. Exclude specific files, file types from copy
    - Evaluation Method: Define files and file types to not be copied, then attempt copy
    - Dependencies:
        - 03. Copy multiple files
    - Priority: Middle
5. Backup directory structure
    - Evaluation Method: Copy directory structure from one location to another.
    - Dependencies:
        - 01. Traverse a directory structure
        - 03. Copy multiple files
        - 04. Exclude specific files, file types from copy
    - Priority: High
6. Pass user credentials and connect to SAMBA (Windows) network shares
    - Evaluation Method: Successfully connect to and read files on SAMBA share
    - Dependencies:
        - 01. Traverse directory structure

- Priority: Middle

7. Check for read/write access on SAMBA shares
   - Evaluation Method: Successfully get directory permissions
   - Dependencies:
     - 06. Pass user credentials and connect to SAMBA (Windows) network shares
   - Priority: Middle
8. Mount filesystems and/or SAMBA shares
   - Evaluation Method: View files on mounted filesystems/shares.
   - Dependencies:
     - 01. Traverse directory structure
     - 06. Pass user credentials and connect to SAMBA (Windows) network shares
   - Priority: Middle
9. Command Line Interface
   - Evaluation Method: Perform successful backup using command line interface.
   - Dependencies:
     - 05. Backup directory structure
     - 08. Mount filesystems and/or SAMBA shares
   - Priority: Middle
10. Multiple Environment Interface
    - Evaluation Method: Use same successful backup interface on Windows, Linux, and Mac OS X.
    - Dependencies:
      - 09. Command Line Interface
    - Priority: Low
11. Get file sizes and modified times
    - Evaluation Method: Successfully get information about a file
    - Dependencies: None
    - Priority: Low
12. Incremental Backups
    - Evaluation Method: Perform multiple successful backups without unnecessary rewrites of already backed up data.

- Dependencies:
  - 11. Get file sizes and modified times
- Priority: If time permits

System Design and Architecture

Entire Section Updated 11/14/10

The Wonder Backup project will follow the following model:

- **Wonder Backup** (Application) which consists of:
  - **Backup Functions** which consists of:
    - getContents(directory)
      - This function returns a dictionary of the contents of the given directory.
    - excludeFiles(files, exclusionPatterns)
      - This function checks the list *files* for anything that may be excluded as defined by *exclusionPatterns*. It then returns a new list without any excluded files.
    - copy(originalFile, newFile)
      - This function copies the file *originalFile* to the location defined by *newFile*.
    - getBackupFiles(sourceDirectory, exclusionPatterns)
      - This function recursively traverses the directory structure from *sourceDirectory*, returning a list of all files that are not excluded by *exclusionPatterns*.
    - makeBackupFolders(sourceDirectory, targetDirectory)
      - This function recreates the folder structure of *sourceDirectory* in *targetDirectory*.
    - targetFilenames(sourceDirectory, targetDirectory, files)
      - This function returns a list of absolute locations of the files that will be created in *targetDirectory*.
  - **File Functions** which consists of:
    - readableSize(n_bytes)
      - This function returns a string of the size defined by *n_bytes* in its highest representation, to the second decimal place.
    - getAttributes(filename)
      - This function returns a dictionary of the file *filename*'s size and modification date.
  - **OS-Specific Functions** which consists of:
    - dirString(directory)
      - This function returns a directory string from *directory* in the proper format according to the host operating system.

- detectOS(directory)
    - This function returns operating system information for the source *directory*. This function can be called without arguments to get information about the host operating system.
- freespace(directory)
    - This function returns the amount of freespace available on *directory*. Correct information is only retrieved in Linux, so it is disabled for other operating systems.
- getCurrentUser()
    - This function returns the string containing the username of the currently logged-in user.
- checkLocation(directory)
    - This function checks if *directory* is an actual location.
- getProfilesFolder(sourceDirectory)
    - This function returns a string containing the location of the user profiles folder for the *sourceDirectory*.
- getProfiles(sourceDirectory)
    - This function returns a list of profiles found on the given *sourceDirectory*.

- **XML Processing Functions** which consists of:
    - readXML(filename)
        - This function returns the ElementTree of the XML file.
    - getLocations(xmldoc, osFamily, osVersion)
        - This function returns a dictionary containing the backup locations from *xmldoc* for *osFamily* and *osVersion*.
    - getExclusions(xmldoc)
        - This function returns a dictionary containing all of the exclusion definitions from *xmldoc*.
- **Command Line Interface Functions** which consists of:
    - Each step of the backup wizard:
        - selectBackupType(messages)
            - The first step of the wizard. The user selects the backup type at this step. The user can select between local, external, and preconfigured backups.
        - selectSource(messages, backupType)

        - The second step of the wizard. At this point, the user enters the location of the source directory to be backed up. This step is skipped if the user selected a local backup.
    - selectTarget(messages)
        - The third step of the wizard. The user enters the target directory to store the backup.
    - selectUser(messages, source)
        - The fourth step of the wizard. The user selects a user to backup from a list of profiles found on the source directory.
    - selectBackupLocations(messages, source, user, locations)
        - The fifth step of the wizard. The user selects which locations to backup. The locations available is determined by the operating system foundon the source directory.
    - selectExclusions(messages, exclusions)
        - The sixth step of the wizard. The user selects exclusions to be applied to the backup, one at a time.
    - startBackup(messages, backupType, sourceDirectory, targetDirectory, user, backupLocations, exclusions)
        - The final step of the wizard. At this point, the backup actually occurs.
  - cliBackup(messages, backupLocations, targetDirectory, exclusionPatterns)
    - This function performs the command-line backup of files defined by *backupLocations* to *targetDirectory*, excluding all files that match the patterns defined by *exclusionPatterns*.
- **Graphical User Interface Functions** which consists of:
  - Each tab of the backup wizard:
    - Tab_Welcome(parent, messages)
        - This is the first tab users encounter. It is simply a welcome screen.
    - Tab_SelectBackupType(parent, messages)
        - In this second tab, the user selects the backup type: local, external, or preconfigured.
    - Tab_SelectSourceLocation(parent, messages)
        - The third tab is used for selected the source location. It does not show up if a local backup type is selected, but it automatically filled in.
    - Tab_SelectTargetLocation(parent, messages)
        - The user selects the target location for the backup in the fourth tab. In

Linux, the amount of free space on the selected location is shown.

- Tab_SelectUser(parent, messages)
  - The fifth tab contains the user selection. The combobox is automatically filled when the source location is selected.
- Tab_SelectLocations(parent, messages, start_location, inbetween)
  - The sixth tab displays the backup location options. Based on the OS version of the source location, certain options are disabled.
- Tab_SelectExclusions(parent, messages, start_location, inbetween)
  - The seventh tab is structured much like the sixth. The user selects the file exclusions to be used.
- Tab_BackupProgress(parent, messages)
  - In the final tab, one last check is made to determine if all necessary information has been entered. If so, the backup begins. Otherwise, an error is displayed and the user cannot continue until this information is selected.

- The interface, WonderGUI().

Implementation

Modified November 14, 2010

- README.txt
- Icon.png
  - The icon that is seen when viewing the About option in the GUI.
- localizations.xml
  - This file contains all the messages, per language for the program.
- wbBackup.py
  - This file contains the backup functions.
- wbFile.py
  - This file contains the file-related functions.
- wbOS.py
  - This file contains operating-system specific functions.
- wbXML.py
  - This file contains the functions for processing XML files.
- wbCLI.py
  - This file contains the functions for the command-line interface.
- wbGUI.py
  - This file contains the functions for the graphical user interface.
- wonderbackup.xml
  - This file contains the information for locations and file exclusions per operating system.

Known Bugs and Other Issues in Wonder Backup 0.3 Beta

- On a 64-bit Mac OS X host, the Python installation must first be set to 32-bit mode.
- On Windows and Mac OS X hosts, the 'statvfs' module is not available. As a result, freespace reports do not work.
- On Windows 7 hosts and Windows 7 guests (including local backups), Documents backup fails due to native symbolic links.
- On Windows and Mac OS X hosts, tabs cannot be hidden, so all tabs are visible at all times.
- On all hosts, preconfigured backups and import/exports are not yet available.

Preliminary Test Plan and Preliminary Test Cases

Test Plan

Purpose

The purpose of Wonder Backup is to perform simple and automated backups given user-selected constraints using a simple interface that is synonymous across all platforms.

Features to be tested

- Command Line Interface, with and without arguments.
- Graphical User Interface, all functionality
- Storing backup on remote locations
- Importing preconfigured settings
- Exporting configured settings
- Incremental and full backups

Approach

The same approach will be used for all operating systems. The same process will be followed for each test case. The program will be started, the case tested, and results recorded.

Environmental Needs

- Operating System
  - Windows XP, 2003, Vista, 2008, 7
  - Mac OS X 10.5, 10.6
  - Linux (Ubuntu 10.10 tested)
- Python
  - Version 2.5+
- wxWidgets
  - Version 2.8+
- wxPython
  - Version 2.8+

Schedule

- November 16, 2010
  - Unit and Integration Testing
- November 23, 2010

    - Operating System-Specific Testing
- November 30, 2010
    - Forced attempts to crash program.

Acceptance criteria

- Program handles all backup jobs appropriately, without crashing or stalling.
- The proper files are backed up, in the proper location.
- No files are deleted during the backup.
- With incremental backups, updated files are updated on the backup location.
- All features function the same across all platforms.

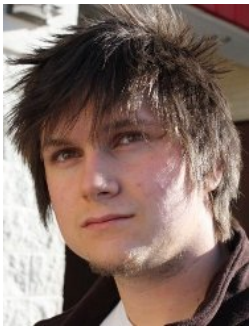
Test Cases

Updated November 14, 2010

- WB-01: Perform a successful backup with command-line version of Wonder Backup under Windows, Linux, and Mac.
    - Steps Involved:
        - Start program by invoking ‘python wbCLI.py’
        - Follow steps of backup program, ensuring that the location the backup is made to is writable.
        - Check if backup was successful.
    - Expected Results:
        - Files that were found in the chosen backup areas will exist in the chosen save directory, excluding the excluded file types
- WB-02: Successfully run a backup from graphical user interface under Windows, Linux, and Mac.
    - Steps Involved:
        - Start the program by invoking ‘python wbGUI.py’
        - Follow steps of backup program, ensuring that the location of the backup is made to is writable.
        - Check if backup was successful.
    - Expected Results:
        - Files that were found in the chosen backup areas will exist in the chosen save directory, excluding the excluded file types

- WB-03: Package Linux version into AppPackage.
  - Steps Involved:
    - Research AppPackages.
    - Gather requirements.
    - Package the files.
    - Test on multiple Linux distributions.
  - Expected Results:
    - Program runs on any distribution without the installation of dependencies.
- WB-04: Package Mac version into .App file.
  - Steps Involved:
    - Research .Apps
    - Package into .App file
    - Test the application on multiple Mac installations.
  - Expected Results:
    - Program runs on multiple Mac installations.
- WB-05: Package Windows version with portable python installation / Compile into .exe
  - Steps Involved:
    - Research portable installations.
    - Compile into single object.
    - Test on multiple Windows installations.
  - Expected Results:
    - Program runs on multiple Windows installations.

About the Author



Sean Davis is a senior Computer & Information Science major at Berea College in Berea, KY. Easily the most stubborn Linux user, he has thrown out his physical Windows partitions in favor of Ubuntu Linux, virtualizing Windows only for work and testing software. He'll soon be graduating and looking for a programming firm (or Linux company) to take him in for his talents in C++ and Python.

Executive Section

To: Dr. Jan Pearce, Project Director

From: Sean Davis

Subject: Wonder Backup

Date: 9/7/2010

Accomplishments: I gathered ideas for the project. I opened the project page for Wonder Backup at http://wonderbackup.sourceforge.net. I also developed a preliminary project logo and icon. Lastly, I compiled the project proposal.

Challenges: Thinking of a project name that was yet unclaimed proved to be a difficult endeavor. Creating the project logo was a tedious process since I used imaging software that I had never encountered. I did manage to overcome both of these challenges.

Time Spent: 2 hours on logo development, 2 hours on project proposal, 10 minutes on the Executive Section.

Goals: Meet with Project Director to plan next phase.

To: Dr. Jan Pearce, Project Director

From: Sean Davis

Subject: Wonder Backup

Date: 9/14/2010

Accomplishments: I completed the Vision and Scope and Preliminary Software Requirements Specifications sections in the Application Development Section.

Challenges: Determining the scope of my project. I feel confident that I should be able to complete a number of requirements in the short amount of time, so considering all that might be necessary for the project was challenging and time consuming.

Time Spent: 0.5 hours on Vision and Scope section, 1.5 hours on Preliminary Software Requirements Specifications section, 10 minutes on the Executive Section.

Goals: I need to properly configure the Mercurial repository on Sourceforge.net, begin code development, and meet with the Project Director.

To: Dr. Jan Pearce, Project Director

From: Sean Davis

Subject: Wonder Backup

Date: 9/21/10

Accomplishments: I reviewed the Software Requirements Specifications and found there to be no issues. I began code development and properly configured the Mercurial repository on Sourceforge.net, submitting my first code and documentation.

Challenges: I had written some preliminary code in the summer, but with the recent version bump to the 2.7 series, it was non-functional. A significant challenge was posed in diagnosing the problem.

Time Spent: 30 minutes Tuesday on code development. 10 minutes Sunday on reviewing the SRS and 40 minutes on code development. One hour on code development Tuesday morning. 10 minutes on the Executive section Tuesday morning. Total time spent this week: 2 hours 30 minutes. Total time spent this term: 6 hours 50 minutes

Goals: Review and design a software architecture for the project and continue software development.

To: Dr. Jan Pearce, Project Director

From: Sean Davis

Subject: Wonder Backup

Date: 9/28/10

Accomplishments: Reviewed code and decided on System Design and Architecture and updated the Application Development Section accordingly.

Challenges: The greatest challenge posed this week was finding an opportunity to work on this segment of the project. While I had considered what to include throughout the week, I did not have an opportunity to follow through until the submission deadline.

Time Spent: Total time spent this week: 1 hour 30 minutes. Total time spent this term: 8 hours 20 minutes.

Goals: Continued development and implementation of the backup program code.

To: Dr. Jan Pearce, Project Director

From: Sean Davis

Subject: Wonder Backup

Date: 10/5/10

Accomplishments: Completed requirements for first working prototype. The program can now successfully backup important file locations to user-specified locations. This is a significant step towards completion of the project.

Challenges: Challenges this week included importing all parts of the project (which proved to be unsuccessful, hence the single all_functions.py) and preparing operating system-specific functions for specific backup locations.

Time Spent: 3 hours dedicated on Sunday, 1 hour on Monday, and an hour Tuesday morning on code development. Half hour Tuesday morning on document updates. Total time spent this week: 5.5 hours. Total time spent to date: 13 hours and 15 minutes.

Goals: Continued development and implementation of the backup program code.

To: Dr. Jan Pearce, Project Director

From: Sean Davis

Subject: Wonder Backup

Date: 10/18/10

Accomplishments: Transitioned configuration to XML. Developed an initial graphical user interface, with proper structure and cross-platform functionality. Reordered code into properly named files. Made several adjustments to documentation.

Challenges: Challenges this week included learning and structuring XML as well as developing a graphical user interface with the wxWidgets toolkit.

Time Spent: 3 hours Thursday, 2 Sunday, and 3 Monday on GUI. 1.75 hours Monday on code fixes and restructuring. 1 hour adjusting documentation as necessary and ten minutes on the Executive Section. Total time spent these two weeks: 10.75 hours. Total time spent to date: 24 hours.

Goals: Debug inconsistencies, develop a functional GUI, further development. Movement to alpha/beta status.

To: Dr. Jan Pearce, Project Director

From: Sean Davis

Subject: Wonder Backup

Date: 10/26/10

Accomplishments: Transitioned most messages and dialogs to XML format. Now messages can be defined in one location and be changed in both the GUI and CLI. Achieved alpha state in command-line interface, allowing for local backups defined in and XML document, and allowing for file type exclusions. Completed XML reading capability.

Challenges: The biggest challenge that was presented this week was manipulating the XML document in order to usable, since there was no documentation. Next, I also had issues with developing the GUI further, so I put my focus on completed the CLI.

Time Spent: 4 hours Sunday on XML reading functions, 2 hours Sunday on GUI work. 3 hours Monday on command-line backup functions. 2 hours Tuesday commenting and organizing code. 30 minutes Tuesday on report updates and 10 minutes Tuesday on Executive section. Total time spent this week: 7 hours, 40 minutes. Total time spent to date: 31 hours, 40 minutes.

Goals: Move remainder of output to XML documents and begin allowing for localization. Complete GUI, possibly with a wizard similar to the command-line backup progress. Work on removable device and network detection. Further test and optimize the program.

To: Dr. Jan Pearce, Project Director

From: Sean Davis

Subject: Wonder Backup

Date: 11/2/10

Accomplishments: Finished ported messages to XML format. Now all messages can be defined from XML document, and localization should be a matter of adding languages to the XML document. Added OS detection to the GUI program. Added event-handling to make tabs interoperate.

Challenges: The greatest challenges that were posed this week included improving and adding functionality to the GUI. Many things are now automated. The one hurdle that still remains is the backup dialog. Upon finishing that, the GUI program will be complete.

Time Spent: Friday evening: 6 until 10, Saturday evening: 5 until 10, Sunday 2 to 8, and Monday 7pm to Tuesday 5am. Total time spent this week: 25 hours. Total time spent to date: 56 hours, 40 minutes.

Goals: Complete GUI. Compile wxWidgets on a Mac. Research packaging options for each operating system. Test the program.

To: Dr. Jan Pearce, Project Director

From: Sean Davis

Subject: Wonder Backup

Date: 11/16/10

Accomplishments: Updated all comments and documentation. Updated file names to a single convention. Completion of the Graphical User Interface. Optimization of underlying code. Updated Command Line Interface. Performed initial testing on each Operating System.

Challenges: Progress information on GUI backup. It was simply enough to make a progress bar for this program, though there was little information on actively updating the progress bar. Other issues were also associated with the GUI backup progress, and they were ultimately resolved over time.

Time Spent: 11/2 through 11/9, 30 hours on code development and optimization for all underlying functions and command line program. 11/10, 2 hours on completion of command line program and 1 hour on updating of comments and documentation. 11/11, 1 hour on documentation update. 11/12, 3 hours on GUI code development. 11/13, 3 hours on GUI code development. 11/14, 2 hours on further updates on documentation and improvement on documentation appearance, 1.5 hours on development of the Test Plan.

Goals: Continued bug testing (and squashing). Practice video demos, and test screen recording software for the demos.