CSC 493 – SENIOR PROJECTS

Wonder Backup

Report 8 on the work of Week 9

Sean Davis

11/2/2010

This document contains information regarding the results of work completed through week 9. At this point, test cases have been included and the graphical user interface is nearly complete. Project updates can be found at http://wonderbackup.sourceforge.net.

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 Typo 10/18/10
    - **Milieu:** # Alphabetized 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/ # Added 10/18/10
            - 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

01 | Traverse a directory structure
- Evaluation Method: List files in deeper subdirectories.
- Dependency: None.
- Priority: High

02 | Copy a single file
- Evaluation Method: Copy a file from one location to another.
- Dependency: None.
- Priority: High

03 | Copy multiple files
- Evaluation Method: Copy multiple files from one location to another.
- Dependency: 02. Ability to copy a single file
- Priority: High

04 | Exclude specific files, filetypes from copy
- Evaluation Method: Define files and filetypes to not be copied, then attempt copy
- Dependency: 03. Copy multiple files

- Priority: Midde

05 | Backup directory structure

- Evaluation Method: Copy directory structure from one location to another.
- Dependencies:
  - 01. Traverse a directory structure
  - 03. Copy multiple files
  - (Optional) 04. Exclude specific files, filetypes from copy
- Priority: High

06 | Pass user credentials and connect to SAMBA (Windows) network shares

- Evaluation Method: Successfully connect to and read files on SAMBA share
- Dependency: 01. Traverse directory structure
- Priority: Middle

07 | 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

08 | 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

09 | 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 OSX.
- 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

The Wonder Backup project will follow the following model:

- Wonder Backup (Application) *which consists of:*
    - Backup Functions *which consists of:* # Modified on November 2, 2010
        - Copy_multiple(origin_directory, target_directory, exluded_filetypes)
            - *This function will copy files from a given* origin_directory *to* target_directory *excluded any* excluded_filetypes.
            - *This function is dependent on* Copy_single(file, location)
                - *This function will copy a single* file *to a given* location.
            - *This function is dependent on* remove_excluded(files, excluded_filetypes).
                - *This function removes files of the given* excluded_filetypes *from the list* files.
        - Get_contents(directory)
            - *This function will gather all contents of a directory and separate them in such a way that subdirectories, files, and symbolic links are divided.*
        - Get_files_for_backup(source_directory, excluded_filetypes)
            - *This function recursively traverses the source_directory structure and returns all files that are not of the excluded_filetypes.*
        - Make_folders_for_backup(source_directory, target_directory)
            - *This function recreates the folder structure of source_directory on target_directory.*
        - New_location_files(source_directory, target_directory, files)
            - *This function returns a list of files with the absolute location set to target_directory.*
    - File Functions *which consists of:*
        - Get_extension(filename) # Modified October 5, 2010
            - *This function gets the extension of the given filename. This is used for exclusion/inclusion purposes.*
        - Get_attributes(file)
            - *This function gets important attributes of a given* file. *These attributes include size and modification date. These will be used in incremental backups as well as possibly estimated backup completion times.*
    - OS-Specific Functions *which consists of:* # Modified October 26, 2010
        - Get_os()

            - *This function gets the current operating system family and version.*
        - Get_logged_in()
            - *This function gets the currently logged in user.*
        - Check_valid_location(location)
            - *This function ensures that a given location is valid to prevent backup offerings that do not actually exist.*
        - Get_os_backup_locations(xml_file)
            - *This function gets all possible backup locations defined in the specified* xml_file.
        - Get_os_users_location(source)
            - *This function returns the location of the user profiles folder for the given source location.*
    - XML Processing Functions *which consists of*: # Modified 10/26/10
        - readXML(filename)
            - *This function reads an XML file and returns the tree information.*
        - Get_backup_locations(tree)
            - *This function returns the backup locations defined in the XML Element Tree.*
        - Simple_locations(xml_tree, family, version)
            - *This function returns the locations from in the xml_tree for the specified operating system family and version.*
        - Get_exclusions(tree)
            - *This function returns the exclusions list as defined by the xml tree.*
        - Get_message(tree)
            - *This function returns the messages as defined by the xml tree.*
        - writeXML(tree, filename)
            - *This function takes the* tree *and writes an XML answer file.*
    - Command Line Interface Functions *which consists of:* # Modified 10/26/10
        - Terminal_output(directory, files)
            - *This function presents the directory information in such a way that is easy to read and understand.*
            - *This function is dependent on helper functions* truncate_directory, terminal_filename, terminal_size, *and* terminal_modified *which help in conversion to strings and presentation.*
        - Other functions found in this region simply run the program.
    - Graphical User Interface Functions *which consists of:* # Modified 10/18/10

- WonderGUI
    - This class will create the interface using wxPython (and therefore wxWidgets).
- Classes for each Tab # Added November 2, 2010
    - These classes will each have special components that will allow for better interactivity (comboboxes, checkboxes, radio buttons) and events that will allow the tabs to interact with one another.
- GuiBackupProgress # Added November 2, 2010
    - This class is a dialog box that will appear to inform the user of the backup progress. As it progresses, the user will be given detailed information as to what is actually happening.

Implementation

- README.rtf, modified 10/18/10
- Icon.png
    - The icon that is seen when viewing the About option in the GUI.
- wb_backup.py
    - This file contains the backup functions.
- wb_file.py
    - This file contains the file-related functions.
- wb_os.py
    - This file contains operating-system specific functions.
- wb_xml.py
    - This file contains the functions for processing XML files.
- wonder_cli.py
    - This file contains the functions for the command-line interface.
- wonder_gui.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 0.1 Alpha

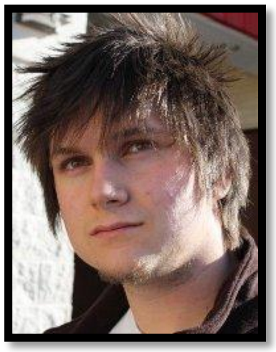
- Graphical User Interface is not entirely complete. All functionality exists except for actually backing up.
- Mac version of the GUI does not yet exist. Until wxWidgets can be compiled, this will not be corrected.
- Local backups must be performed by selecting the root of the local hard drive.
- Terminal output is currently disabled.
- Filename-based exclusions are not currently available.
- Windows version does not currently work due to a flaw in directory slashes (/ vs. \ )

Preliminary Test Plan and Preliminary Test Cases # Added November 2, 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 wonder_cli.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 filetypes.
- WB-02: Successfully compile wxWidgets on Mac OS X
    - Steps Involved:
        - Download source.
        - Compile
        - Check Compilation output
    - Expected Results:
        - Compilation was successful
- WB-03: Successfully run a backup from graphical user interface under Windows, Linux, and Mac.
    - Requirements:
        - WB-02 (Successfully compile wxWidgets on Mac OS X)
    - Steps Involved:
        - Start the program by invoking 'python wonder_gui.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 filetypes.
- WB-04: 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-05: 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-06: 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 filetype 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.