CSC 493 – SENIOR PROJECTS

Wonder Backup

Report 5 on the work of Week 5

Sean Davis

10/5/2010

This document contains information regarding the results of work completed through weeks 1 through 4. This work includes the Project Concept Proposal, Scope & Preliminary Requirements Specification, System Design and Architecture and the work behind these processes, and beginning implementation. 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 is [tentatively] licensed under the Python Software Foundation (PSF) license.
    - **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 web interface, command line, or through an answer file.
    - **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 Windows network shares.
    - **Milieu:**
        - 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++.
        - 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.
    - **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:**

    - Python Programming Language 2.7
        - http://www.python.org/
    - Samba Windows Interoperability Suite
        - http://www.samba.org/
- **Challenges:**
    - Python versioning on Unix systems with Python pre-installed.
    - Web Interface
    - 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.
    - Web interface is functional.
    - 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:*
    - 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.*
  - 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 4, 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_user_locations(username)
      - *This function gets the possible backup locations for the currently logged in user.*

    - Terminal Output Functions *which consists of:*
        - 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.*
    - Graphical User Interface Functions…
        - *Which are as of yet unknown.*

Implementation

- README.rtf, completed October 5, 2010
- demo.py, completed October 5, 2010
    - This file contains the driver functions for this initial demo.
- all_functions.py, completed October 5, 2010
    - This file currently contains all the functions that are normally contained in file_functions.py, backup_functions.py, output_functions.py, and os_specific.py. For whatever reason, when trying to run the program importing each piece was not working properly, so they were contained in a single file for sake of operation.

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.