Deploying or Upgrading the “surveyprocess” Software

Prologue:
At the time of this writing (February, 2016), I'm assuming that everyone is using Windows 7. Windows 10 is out, but our versions of both Microstation/Geopak and AutoCAD are not supported on Windows 10.

I writing this mainly as a checklist for myself. I hope to add enough detail that someone can follow behind me. The Introduction should be all a typical user needs to know.

Introduction:
surveyprocess is a set of perl scripts (programs) used to convert raw survey ASCII files into processed ASCII files ready for loading into Geopak using the IDOT standard codes. The raw files (in most cases) are:

1. created using MPS point code listing and following MPS line coding procedures.
2. typically downloaded from the data collector as a comma delimited file in 'point number, northing, easting, elevation, description' format. The description includes the Point Code, the Line Code (if any) and the Comment (if any).

There are two scripts that are run against the ASCII files to produce the Geopak-ready files (Note that Geopak must also be customized for the files to be processed correctly). The two scripts are “checkin.pl” and “process.pl”. The scripts are written in the “perl” programming language.

“checkin.pl” verifies that the Point Codes are a part of the MPS coding scheme. “checkin” takes a file with the '.txt' extension as input and produces a file with the same name but with a '.csv' extension.

“process.pl” takes the file with the '.csv' extension as input, makes the necessary conversions and outputs one or more files with the '.cor' extension. This '.cor' file is imported into Geopak.

The “checkin.pl” and “process.pl” are simple programs that are run from the 'Command Prompt' (cmd.exe) in Windows. The 'Command Prompt' is similar to the old DOS prompt. To easily access the 'Command Prompt' for this purpose, use the 'Windows Explorer' program to browse to the folder that contains the ASCII file you want to process; hold down the shift key and at the same time right-click on the folder name; a pop-up window will appear; select “Open command window here”. Then the programs can be run with the commands:

checkin filename.txt

and

process filename.csv

Essentially that is all most users need to know about how the system operates:

1 Open up the 'Command Prompt' window in the folder that contains the files to be processed.
2 Run “checkin” on the “.txt” file.
3 Check the “.csv” file for errors
4 Make corrections to the “.txt” file until the “.csv” file passes “checkin” without errors.
5 Run “process” on the “.csv” file.
6 Import the “.cor” file(s) into Geopak.

What Else is Going On:

Most of the time the above procedure is all you need to know, however, sometimes we make changes to “checkin.pl” or “process.pl”. We might want to add or remove some codes. We might want convert the data to a different client's format. If I change a code on my machine, what happens on everyone's machine? If I change to convert our codes to a different Non-IDOT client's format, how does that change the original “checkin.pl” and “process.pl”? I'm working on the processing the files for SS4 (OpenRoads SS4), how do we switch back and forth from the original system to SS4?

To help solve these problems we are using a version control system named “git”. “git” is a bit of a pain in the ass, but it handles the above problems very well. It is also very safe. You don't have to worry about screwing up the perl scripts by making changes to them. You don't have to worry about how someone else is processing at the moment.

”git” is a distributed system. That means every machine has it's own copy of surveyprocess. The copies are called repositories, “repos” for short. The “repos” contain ALL of the versions of surveyprocess; all of the old versions and all of the conversions-to-client-formats that we have done so far.

There is a master copy, usually referred to as “origin”. “origin” is located on a virtual server in our Morton office. The server's IP address is 192.168.50.150. The copies are cloned from “origin”. And it is easy to get back to the original state or update to the latest version by pulling the information from “origin”.

The server runs a software package named “gitolite”. “gitolite” maintains the “origin” repository and other repositories on the server. It maintains a list of permissions, so that all authorized personnel can pull any changes, certain people can make changes to the repositories and it ensures that there is no un-authorized access.

There is a lot more to “git” and “gitolite” and how the backend of this all works, but the main points are:

1 Every machine has it's own copy of the repository.
2 Changes can be made.
3 Changes can be incorporated into the origin.
4 You can always get back to the origin.
5 There is a backup copy of each user's repository on the gitolite server.

New Installation:

If this machine is going to be accessible remotely then IT360 needs to set the permissions before this machine can be accessed. IT360 will need the name of the machine to set that up.

New installations require some software to be installed. You need to know if it is a 32 or 64 bit Operating System to download the correct program. You can see this by clicking on the Start Menu, right-click on Computer and then click Properties.

There are two programs that need to be installed; “Perl” and “Git for Windows”. Both programs are opensource and free to use. You can use the default settings when installing these, just say OK to everything.

“Perl” is the scripting language that the checkin.pl and process.pl are written in. The 32-bit or 64-bit versions of Perl can be downloaded and installed from:

HTTP://strawberryperl.com/

“Git for Windows” is the version control software that we use to maintain the surveyprocess files. It lets us make additions, corrections, or modifications to the surveyprocess files with out fear of screwing-up another machine's set-up. It also provides “easy” ways to get back to where we were before we started changing anything. It can be downloaded and installed from:

https://git-for-windows.github.io

Git Bash:

We work with git through a command-prompt-type-window in a program named “Git Bash”. “Git Bash” is not the same as the “Command Prompt” that we use to run the perl scripts. It is based on a UNIX operating system, not Windows. That really doesn't matter to us, except that because it is UNIX, **case matters**. The file names: topo.txt, TOPO.TXT, Topo.txt, tOpO.tXt and ToPo.TxT are all the same in Windows. In UNIX they are five completely different files. The same goes for commands. Everything in “Git Bash” is case sensitive.

You can access “Git Bash” from the Start Menu – All Programs – Git – Git Bash.

“Git Bash” is probably not set to open up in the right folder for our purposes. Go to start it up, but instead of clicking on it, right-click on it and select Properties.

In the “Target:” field remove the text:

--cd-to-home

Change the “Start in:” field to read:

C:\git-repos\surveyprocess

New User:
There are a few items that need to be addressed for a new User, both on the machine the User is going to use and on the gitolite server. The User name for all of this, is the same as the regular username of the Company; first initial – last name. John Smith's username is jsmith. When you see USERNAME, insert the correct username.

On the machine the User is going to be using we need to set up the authentication for the User. This is needed to access the gitolite server. Gitolite allows certain users to do certain things. Regular users can pull the origin and make changes to their backup copy of the repository. Administrative users can do other things, including looking at each users backup repository, administrating the server and making changes to origin.

Gitolite uses a popular encryption method to guarantee that users are allowed the proper access. We need to set this up for each user. This is like a password, but once it is setup, it is not necessary to type in a password. The server can just tell who the user is and what access rights that they have. For the record, we are using the SSH protocol and public-key cryptography.

To set this up we need to add a folder to the user's machine. So, **logged in as the User**, open “Git Bash” and move to where we want to add the folder; type:

cd ~

that's the tilde, Shift and the key below ESC. “cd” means change directory or folder. The tilde is short-hand for “C:\users\USERNAME\”.

From a previous install, there may be a .ssh folder already there. Test this by typing:

ls -al .ssh

If there are files listed, you can skip to the next section “Already Got Keys”. If not, then create the folder:

mkdir .ssh

mkdir means “make directory”. A “directory” is what “folders” used to be called in Windows. The period before the ssh is important.

Then move into that folder:

cd .ssh

Then create the key pair. There is a public key and a private key. The public key gets copied to the gitolite server and private key stays in the .ssh folder and only in the .ssh folder. When git needs to do something on the gitolite server it sends the private key to the server. The server checks to see if they “work” together and allows or declines access. Type:

ssh-keygen -f USERNAME

When asked to enter to “Enter passphrase”, leave it blank and press enter twice.

This creates two key files, “USERNAME” and “USERNAME.pub”. “USERNAME” is the private key which stays where it is and “USERNAME.pub” is the key you can email to the gitolite administrator.

Next we need to create a configuration file in the same folder with the name “config”. You can do this with Notepad, UltraEdit or any other plain text editor; not Word or any other word processor. This is what the file should look like. Remember to replace the USERNAME.

```
Host USERNAME
        User          g_admin
        Hostname      192.168.50.150
        Port          22
        IdentityFile  ~/.ssh/USERNAME
```

Already Got Keys:
If the .ssh folder is already there, then it probably contains two files, “id_rsa” and “id_rsa.pub”. Make a copy of “id_rsa” and name it “USERNAME”. Make a copy of “id_rsa.pub” and name it “USERNAME.pub”.

Now go back up the text a few paragraphs and email the “USERNAME.pub” file and

create the “config” file.

Logout as User:
Once the key files are created, the “USERNAME.pub” key sent to the admin and the “config” file created, we don't need to be logged in as the user for now.

Changing the PATH:
Still on the new machine. We need to change the PATH. The PATH is a list of folders that Windows uses to search for files. For instance, if we type “checkin” into the command prompt, Windows searches through the PATH list until it finds a file that it can run named “checkin”.

From the “Start Menu”, select “Computer”.

Select “System Properties” near the top of the next window.

From the next window select “Advanced system setting” from the left side.

From the “System Properties” window, select the “Advanced Tab” and then the “Environment Variables” near the bottom.

On the “Environment Variables” window, we will use the bottom section, “System variables”. Using the slider on the right, go down until you see the word “Path”. Double click on it.

The “Variable value:” is the list that Windows looks through. The individual paths typically start with “C:” and are separated with a semi-colon “;”. Be careful here. Changing the wrong thing could screw-up other programs on the machine. There are two things to do:

1 Add the following path to the list:
C:\git-repos\surveyprocess\BatchFiles;
2 Delete any old path statements referencing “surveyprocess”.

Click OK and exit all the windows.

Administrator's Turn:
Now the gitolite administrator needs to add the public key into gitolite, configure gitolite for the new user and clone and set-up the repository and remotes; see “Add new USER to the Gitolite Server” and “Clone and Setup Repositories and Remotes” below. When that is done we need to log back into the machine and do the next item.

User Logs Back In:
We have been substituting the user's computer username when we see USERNAME. In the following sections we will continue to do that and in addition we will be

substituting the user's first and last, lower case, initials in place of USER-INITIALS. So, “js” for John Smith.

User logs back in. Starts “Git-Bash”. And enters the following command:

git push –all USER-INITIALS_public

So for John Smith, that is:

git push –all js_public

That is 2 dashes in front of the “all”.

And that is the initial setup for the User. The User should be able to follow the Introduction above to use “surveyprocess” to process survey files. As mentioned, the User can make changes and switch into other modes (branches), like the new SS4 conversion or conversion to a different Non-IDOT client's format, but I'll need to write a different paper about that. The rest of this paper is for the gitolite administrator.

Add new USER to the Gitolite Server:

Three steps:

1. Copy the User's public key in the gitolite-admin/keydir folder. This file should be named USERNAME.pub. If it came as id_rsa.pub rename it.
2. Modify the gitolite-admin/conf/gitolite.conf file. Use a text editor, not a word processor.
    1. Add the user to the list of @sp_users.
    2. Add a repository for the User to the list of other Users; @sp_repos = USERNAME.
    3. Add permissions for the new repo:

        repo sp_USERNAME
        RW+ = USERNAME
3. Push the changes to the gitolite server. You need to be in the gitolite-admin directory (Lance – remember this is on your VB-arch machine) to do this:

    git status
    git add -A
    git commit -m “add USERNAME”
    git push

Clone & Set-up Repository and Remotes:

This next section can be done using a remote login on the new User's machine. This is a part of my checklist. If someone else is administering this, it will need to be tweaked a bit.

Copy the “lance” & “lvinsel” keys and the “config” file into C:\users\lvinsel\.ssh folder. Use the server (T:) as an intermediate copy location and remember to delete the files from the server.

Open “Git Bash” and enter the following commands:

cd c:

mkdir git-repos

cd git-repos

git clone mps01:surveyprocess.git

cd surveyprocess

git remote remove origin

git remote add origin USERNAME:surveyprocess.git

So for John Smith, that is:
git remote add origin jsmith:surveyprocess.git

git remote add USER-INITIALS_public USERNAME:sp_USERNAME.git

And again for our buddy John:
git remote add js_public jsmith:sp_jsmith.git

After the User finishes the section “User Logs Back In:”, above, that should be it for the setup for a new machine/user.

Setup the Gitolite Server:
The gitolite server is running on a virtual machine on the VMware ESXi server in the Morton office. The IP address of the server is 192.168.50.150. The server is running the CentOS version 7 Linux operating system.

Gitolite was installed using http://www.bigfastblog.com/gitolite-installation-step-by-step as a guide. “g_admin” was used as the administration user, instead of “gitolite” as used in the instructions. A scanned copy of my installation notes are stored in the master branch under OtherInfo\gitolite_install_notes.pdf.

Administrating the Gitolite Server:

Once the server is setup, gitolite uses git to manage the administration. There is a git repository on the administrator's machine named gitolite-admin. The files in this repo are modified and then pushed to the gitolite server to complete the process. So the basic process is:

1. Add or delete the public key files.
2. Or make changes to the “gitolite-conf” file.
3. *git add -A*
4. *git commit -m 'commit message'*

The gitolite-admin repo has two directories that we are concerned with. The first one is “keydir”. This is where the admin places all of the public ssh keys for all of the users. The public keys need to be named USERNAME.pub not the generic id_rsa.pub; or in the case of the admin, ADMINUSER.pub.

The other directory is “conf”, which contains the “gitolite.conf” file. “gitolite.conf” is a text file, so you can use UltraEdit to modify it.

Add a New User to the Server:
Move the User's public key into the “keydir” directory.

In the “gitolite.conf”, add a new line under the “# regular users” section. Just follow suit.

Add a new line under the “# surveyprocess local repos”. Again, just follow suit. This is the copy of the Users repo. If the User makes change to surveyprocess, this is where those changes get pushed to. Then the admin can look at the changes and decide if they should be integrated into the origin.

Save, exit and do steps 3 and 4 from above and you are done with setting up the new user in gitolite.

Add a New Admin to Gitolite:
To add a new administrator the current admin needs to change the permissions for the new admin so that they can access the gitolite-admin repo. We need another USERNAME for the administration user. The admin will have two logins and two different keys, one for the regular use and one for administration only purposes. As an example, I'm using “lance” for the admin and then I have “lvinsel” for the regular user.

Move the Admin's public key, “ADMINUSER.pub”, into the “keydir” directory. You need to create this key on the New Admin's machine; see the next section for that.

In the “gitolite.conf” file, under the section labeled “# teamlead Users”, add a line:

@sp_users = ADMINUSER

After the line “repo gitolite-admin”, add ADMINUSER to the end of the line, like:

RW+ = @admin lance ADMINUSER

Save, exit and do steps 3 and 4, from above and you are done with getting the gitolite server setup for the new administrator.

Setup the New Admin's Machine:
Before you can add the new key, in the previous section, you need to create it. Open “Git Bash” and move to the .ssh directory:

cd ~/.ssh

ssh-keygen -f ADMINUSER

In my case this was:
ssh-keygen -f lance

The ADMINUSER.pub is the file that needs to be copied in the Current Admin's “gitolite-admin/keydir” folder.

In the same folder, “c:\users\USERNAME\.ssh”, is a file named “config” that needs to modified. Using UltraEdit or some other text editor, add the following to the end of the file:

```
Host ADMINUSER
    User         g_admin
    Hostname     192.168.50.150
    Port         22
    IdentityFile ~/.ssh/ADMINUSER
```

So once everything above is done, change into the git-repos folder with “Git Bash”:

cd c:\git-repos

git clone ADMINUSER:gitolite-admin

This pulls the gitolite-admin repo into the folder. Then you can change into the new repo:

cd gitolite-admin

and make changes to the “keydir” and “conf” folders and other administration functions and then push the changes to the gitolite server.