WORK AREA FOR THE ASSIGNMENTS

The work for all assignments should be done in a special directory beneath your home directory on *hills*. This directory must be named **cs160a**. Within **cs160a** you must create a directory for each assignment as needed. The directory must be named the same as the assignment, e.g., files for Assignment 1 (asmt01) must be in the directory **asmt01**, or **cs160a/asmt01** relative to your home directory. You should leave all files for the assignment in place until after you have verified that your grade has been recorded. I suggest that you leave all your files in place until the end of the semester. You may find them useful both for review and for future assignments. (The procedure for creating these directories is detailed further in the handout for Assignment 1.)

FACILITATING GRADING

You are responsible to do your assignments so that I can readily follow what you have done. If I can't *readily* follow your assignment, I cant grade it. Resubmission of ungraded assignments are permitted at my discretion, and will incur a penalty.

To facilitate this, you must follow certain rules.

RULES FOR THE ASSIGNMENT EXTERIOR

1. All parts of your assignment *must* be attached together when you hand them in. If you print your assignment, staple it together *so that the text is not obscured.*
2. Each part of your assignment must have your name, **your hills login** and the class day and time and the assignment number plainly visible on the front page. Here is an example of what is required for a student in a Tues/Thurs 11am section:

```
Jane Doe
CS160A. Evening Section
Assignment One
TR 11am
jdoe
```

RULES FOR THE ASSIGNMENT ITSELF

Most assignments will be done as a **script** session. General information on using **script** for assignments is covered in the following section entitled *Preparing Assignments with Script*.

DESCRIPTIVE HEADER

Each **script** session must be placed in a separate file. I highly suggest name your script output files - if you use the default name of **typescript**, **script** will silently overwrite any existing **typescript** file which may be for another part of the assignment. I suggest you choose a standard naming scheme. ***Make sure that whatever name you choose is not the name of an existing file. Remember, Unix doesn't warn you if you do this - it simply erases the file with the same name and starts a new one!***

Each **script** session must begin with a standard descriptive header. This consists of:

1. A comment containing the assignment number, part number (if there is one) and your name.
2. The date. This is inserted by using the **date** command.
3. The current directory. This is inserted by the **pwd** command.
4. Your login id. This is inserted by either of the commands **id** or **whoami**

Example:

You are doing Assignment Two. You should begin the **script** program and name the script file, using a command such as

```
$ script asmt02.script
```

This starts **script** and names the script file **asmt02.script**

If you use the name of an existing file here the file will be overwritten by script.
Unix will respond:

```
Script started, file is asmt02.script
```

The first lines you should type after beginning the **script** program are

```
$
$ #******* Assignment Two Your-Name-Here
$
$ date
```

<the date will appear here>

```
$ pwd
```

<the name of the current directory will appear here>

```
$ id
```

<your user name and other information will appear here>

STEP MARKERS

Some parts of the assignments are fairly long. Since the **script** output consists of Unix commands, these can be difficult to follow. To allow you to make mistakes and me to follow the result, these assignments are divided into distinct steps. Each step is numbered, and literally contains the word **Step**

This procedure requires step markers	This procedure does not
Step 1. blah, blah, blah	1. **blah, blah, blah**
Step 2. blah, blah, blah ...	2. **blah, blah, blah ...**

If a procedure contains **Step**s, as does the one on the left above, you must indicate where the output of each step begins in your **script** session by inserting a *Step marker*. If the procedure contains a numbered list without the word **Step**, like that on the right, you do not have to insert Step markers into the **script** output.

Step markers are inserted directly into the **script** session by you typing at the keyboard. To keep the grader in a good mood while grading (which is to your benefit), observe the following when annotating with step markers:

1. highlight them by placing a blank line before and after the marker and add some leading asterisks.
2. be sure to indicate the number of the Step that is being started.

When you begin a new step of the assignment, which is indicated in the assignment instructions as **Step 2.**, for example, you should insert comments such as

```
$
$ # Step 2
$
```

prior to the commands to perform the step itself. (The **#** begins a comment. This indicates that the remainder of the line is an annotation. If you do not start the line with a **#** the shell will try to interpret the line as a command.)
The example at the end of this handout illustrates this.

PREPARING ASSIGNMENTS WITH **script**

Assignments in this class involve typing a sequence of Unix commands to the shell command line to perform some task(s). To allow me to see what the commands you use as well as the results that derive you will run a Unix command called **script** that copies everything you type and places it in a file as well as on the screen. This seems simple, but there is one thing that is different about using **script** than you are used to: script records keystrokes - it copies everything you type character by character. Thus, if you make a mistake and type a backspace, the erroneous character will be placed in the file, then the backspace, then the replacement character! Since you wont be printing your script file, it is fine

so long as it displays correctly on the screen after you complete it. To make sure this is true, limit corrections when typing commands to the backspace character and **do not use the arrow or TAB keys for command-line editing or history substitutions.**

There are a handful of other things that you must be careful of when you are using **script**:

1. script names its output file **typescript** or whatever other name you give it:

```
script                  # the output file is typescript
script asmt01           # the output file is asmt01
```

   As with other Unix commands, **script** overwrites its output file. Be sure that you do not give it the name of a file you want! (A common mistake when students are writing a shell script is to give **script** the name of the shell script. It dutifully overwrites their shell script!)
2. do not use command-line editing (the up-arrow) when you are typing commands to **script**. Command-line editing overtypes the line, making the result unreadable when it is printed.
3. remember that you are using **script**, and don't forget to exit **script** when you are done. To exit **script**, simply type **exit**. It will respond that you are leaving **script** (see the example)
4. Your **script** sessions should be simple. Practice the assignment before you use **script**. Once you know the commands you need, start over using **script** and record the final version in the **script** session to hand in.
5. If you must explain something while you are using **script**, simply type them to the shell as comments. (see the examples above)
6. don't edit your **script** output file, and if you must cut and paste it, do it carefully. If your commands don't make sense, the assignment can't be graded.
7. Never use a screen-oriented program, such as **vi**, **nano**, **pico** or **more**, when you are using **script**.
8. If you must interrupt your script session you can add to the end of the output file (append to it) by using **script -a** *scriptfilename*. If you do not use the **-a** option, your script file will be overwritten.
9. Never display your **script** output file using **cat** or **more** when you are creating it. This will create an infinite loop. (think about it!) You should only display the file after you have exited **script**.

*Handing in the **script** output file*

Printing from hills is no longer available in the ACRC. Because of this, you will transfer a copy of your script output file to me on hills using the procedure below. I will grade it online. **All you have to hand in for the script session is the cover sheet (see under Assignment Exterior above) for me to write comments on, with the notation *script session transferred*.**

After exiting script, review the **script** output file and make sure you are satisfied with it by examining it with **cat** or **more.** Once you are satisfied with the **script** output file, follow these steps to transfer it:

1. *Your **output-file must be named as indicated in the assignment!*** If it is not, you must rename it. For example, suppose your script output file is named **typescript** and it must be named **asmt01.script.** Simply rename it:

```
mv typescript  asmt01.script
```

2. run the **transfer** program. To do this, you must be in the same directory as the file you want to transfer (not the **share** directory that the **transfer** program will create for you). Then, using the filename from our example above, run

```
/pub/cs/gboyd/cs160a/bin/transfer asmt01.script
```

   This will copy your file to a special directory **~/cs160a/share** where I can retrieve it later. I will try to copy your files to my area during the class in which the assignment is due. (If I forget, remind me).

The following example puts all of this together:

AN EXAMPLE SIMPLE ASSIGNMENT SESSION

Suppose you are Greg Boyd doing Assignment X, which has the following steps.

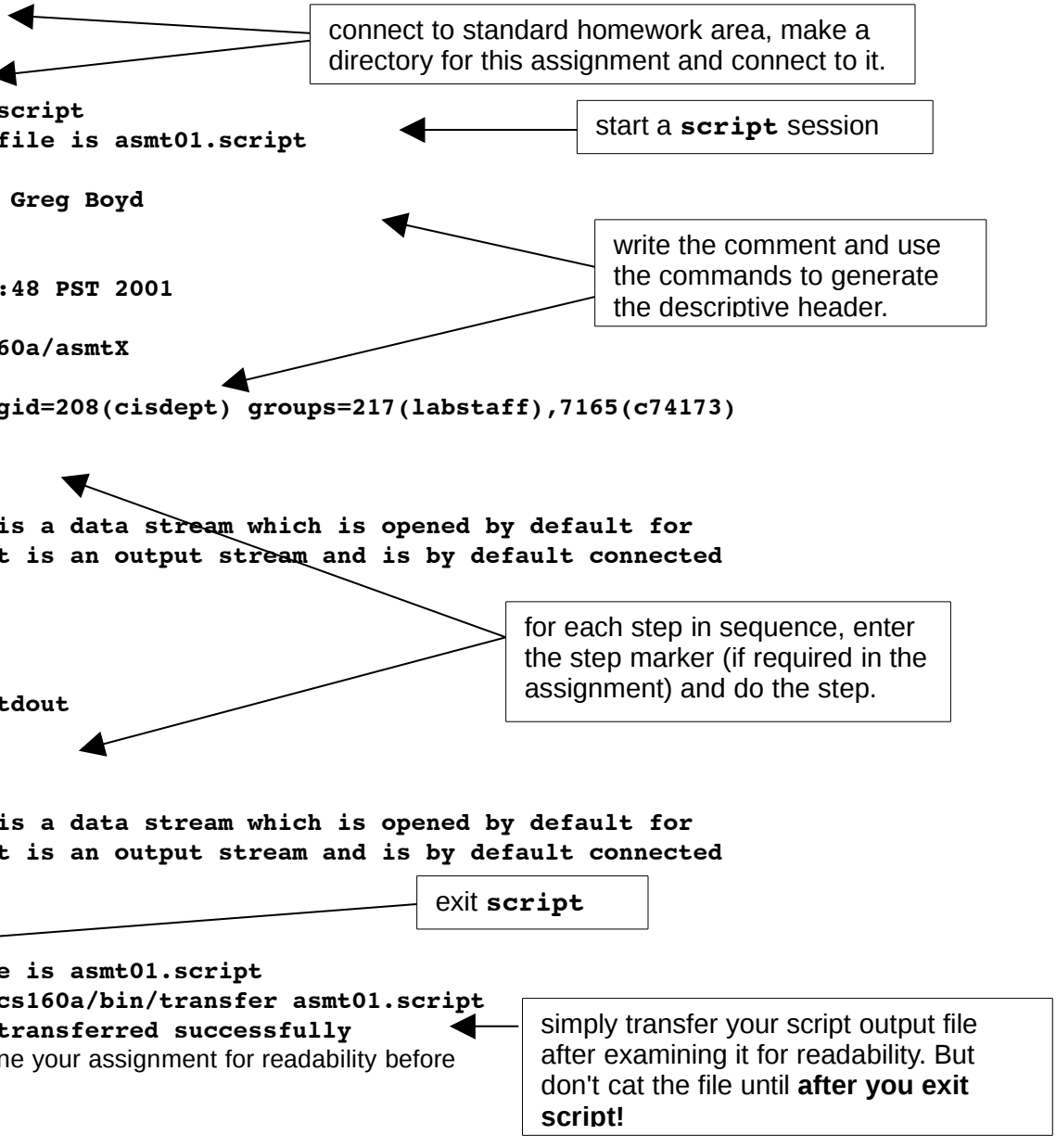
Step 1. Use **cat** to create a file named **stdout** which has the answer to the question *What is standard output?*

Step 2. List the contents of the current directory on the screen

Step 3. Use **cat** to display the contents of the file you created in step 1.

Below is a terminal session (with annotations added) that shows the commands you would have to use to do this part of the assignment. It includes starting a **script** session, inputting the standard descriptive header, performing the three steps, exiting **script** and transferring the results.

```
$ cd ~/cs160a
$ mkdir asmtX
$ cd asmtX
$ script asmt01.script
Script started, file is asmt01.script
$
$ # Assignment X Greg Boyd
$
$ date
Thu Jan  4 17:42:48 PST 2001
$ pwd
/users/gboyd/cs160a/asmtX
$ id
uid=3496(gboyd) gid=208(cisdept) groups=217(labstaff),7165(c74173)
$
$ # Step 1
$ cat > stdout
Standard output is a data stream which is opened by default for
every process. It is an output stream and is by default connected
to the monitor.
$
$ # Step 2
$ ls
asmt01.script  stdout
$
$ # Step 3
$ cat stdout
Standard output is a data stream which is opened by default for
every process. It is an output stream and is by default connected
to the monitor.
$
$ exit
Script done, file is asmt01.script
$ /pub/cs/gboyd/cs160a/bin/transfer asmt01.script
'asmt01.script' transferred successfully
```



Make sure you examine your assignment for readability before transferring it.