Assignment 1 – time24

CS244

Due: Tuesday, September 17, 2013, 11:59 PM

Late penalties will be as described in the syllabus.

Overview

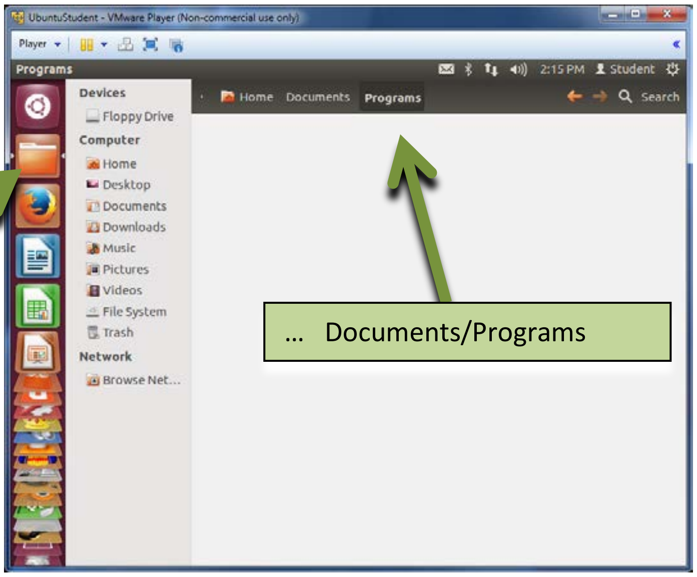
This assignment is for students in CS244 sections with instructor: Brent M. Dingle, Ph.D.
Assignments for sections with other instructors may be different.

Implement the specified changes to the time24 code that should be downloaded from Learn@UW-Stout under Content (aka D2L, Content folder for the course)

Details

A. Create a folder to keep everything for this assignment

Log into your Ubuntu Linux Virtual Machine.
Open the folder browser.
Go to Documents
Go to Programs
(or create a folder named Programs if needed)



B. Download the starter files

Open FireFox, if not already open
Log into the CS244 course page in D2L
Go to Content
Locate the Assignment 1 folder
(likely the same place you found this document)

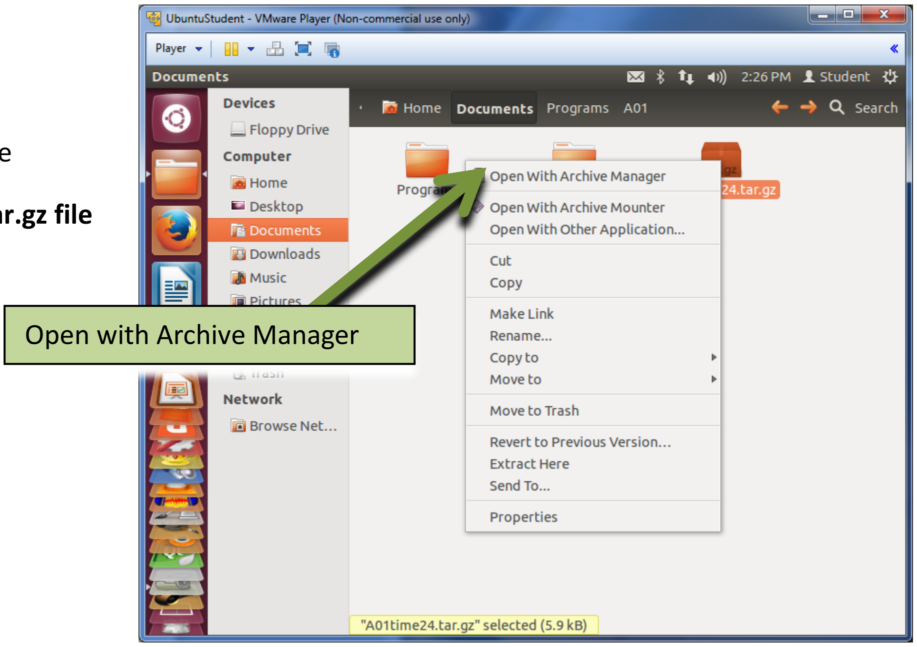
Download the A01time24.tar.gz file

(keep track of where it gets saved to when you download it)
Logout of D2L
Close FireFox

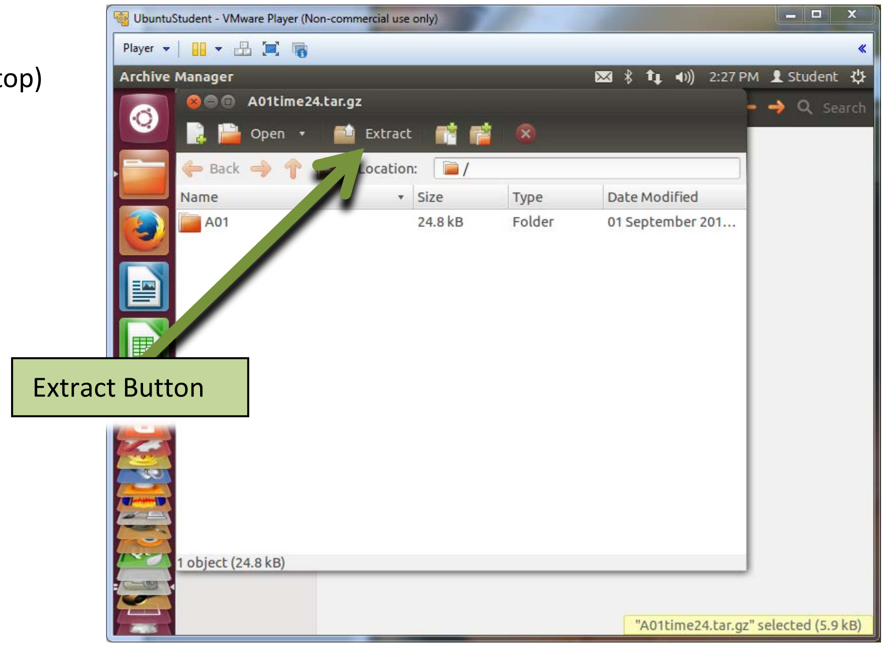
C. Extract the files from the archive

Right click on the A01time24.tar.gz file

Open with Archive Manager


UbuntuStudent - VMware Player (Non-commercial use only)
Documents
Devices
Floppy Drive
Computer
Home
Desktop
Documents
Downloads
Music
Pictures
Network
Browse Net...
Home Documents Programs A01
Open With Archive Manager
Open With Archive Mounter
Open With Other Application...
Cut
Copy
Make Link
Rename...
Copy to
Move to
Move to Trash
Revert to Previous Version...
Extract Here
Send To...
Properties
"A01time24.tar.gz" selected (5.9 kB)
Open with Archive Manager

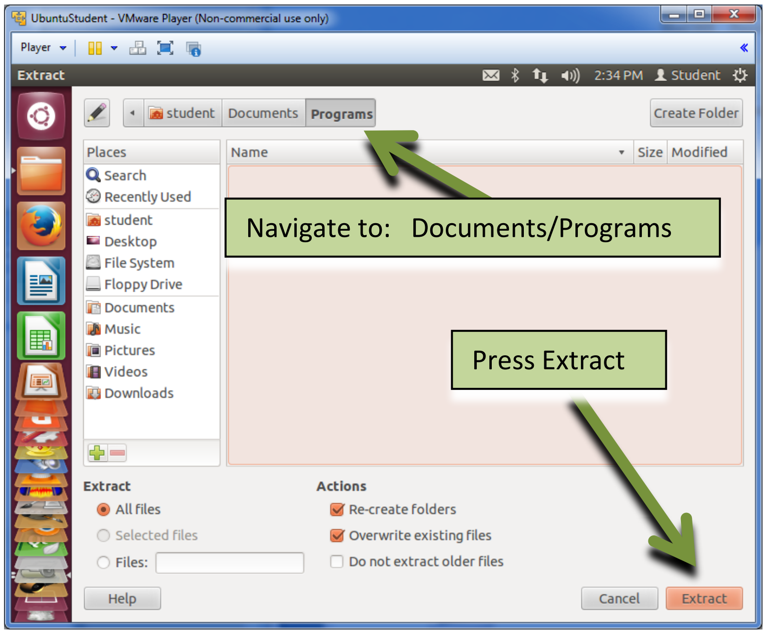
Click the Extract Button (near the top)



Navigate to your Documents folder
Navigate to your Programs folder

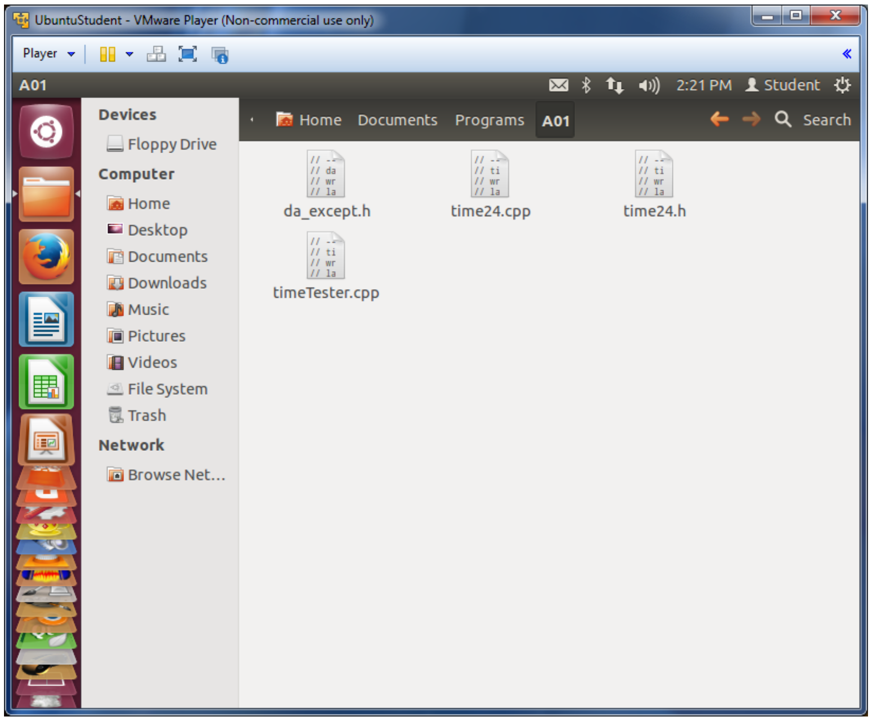
And extract the content to there

This will create a new folder named A01
with the starter source files for
this assignment



You should now have the following files in your newly created A01 folder

time24.h
time24.cpp
timeTester.cpp
da_except.h



You will be modifying these files for this assignment and turning in the modified files.

You may edit them using the editor of your choice.

You may compile them using g++ in whatever manner best suits you.

HOWEVER, **<u>for grading purposes</u>** the following command line will be used in a terminal window to compile them. So you should verify it works before you turn in your assignment files.

g++ timeTester.cpp time24.cpp –o timeTest.exe

To do so:

Open a terminal window

Navigate to your A01 folder (possibly as follows, depending on where you created it)

```
cd Documents
cd Programs
cd A01
```

Compile and link the code into an executable file named timeTest.exe

```
g++ timeTester.cpp time24.cpp –o timeTest.exe
```

This should succeed without warnings (assuming no changes to the files have been made)

Type

```
ls
```

and you should now see a newly created timeTest.exe file

Type

```
./timeTest.exe
```

and the program should execute

Random note:

“(public) member functions” and “methods” mean pretty much the same thing in general use.

Note: Directions in this document take priority over those in the provided starting source code

You should complete the following tasks in order, as some expect their predecessors to be done.
Example: Task 4, part c uses task 3's t3 variable.

Task 1 (15 points)

In each source file update the header comments to include your name and the due date of the assignment. Also update the description comments as indicated using appropriate verbiage.
Suggested: compile and run the program before moving to the next task

Task 2 (10 points)

Rewrite the constructor implementation of time24 without the initialization list.
After you make your changes, verify the initial default value of variables of type time24 remain 0:00. No additional code needed for verification. The output of task 3 should be sufficient.

Task 3 (16 points)

In timeTester.cpp create variables t1, t2, t3, and t4 of type time24
with the following initial values:

initialize t1 to have a value of 6:38
initialize t2 to have a value of 8:00 (init with only ONE integer)
initialize t3 to be 11:30
do not explicitly specify an initial value for t4

use the member function writeTime() to output the values of each variable.
Suggested: compile and run the program before moving to the next task

Task 4 (19 points)

Add a new public function to the time24 class.
Name this function subtractHour(unsigned int h)
Its purpose is to subtract the specified number of hours from the variable's current time
Example: if t1 is 4:37 then calling t1.subtractHour(2) results in t1 being 2:37

Part a

In time24.h properly declare the new function and include pre- and post- condition comments, such as: it takes a non-negative integer to decrease the hour of the time24 object.

Part b

Implement the method in time24.cpp as a member function
Be sure to normalize the result as is done in the addTime method

Part c

Using the existing t3 variable in timeTester.cpp,
subtract 3 hours from it using the subtractHour method

Part d

Output t3's new value using cout and again using writeTime
Each value should be on its own line.

Suggested: compile and run the program before moving to the next task

Task 5 (11 points)

Compare the results of the duration method with the subtraction operator -

Part a

Store the result of t1.duration(t2) in a time24 variable named t5
This must be done in a try block due to range error

Part b

Store the result of t2 – t1 in a time24 variable named t6

Part c

if t5 is equal to t6 output “duration is the same as subtraction\n”
else output “duration is NOT the same as subtraction\n”

Suggested: compile and run the program before moving to the next task

Task 6 (11 points)

Use the operators << and >> with time24 objects

Part a

Prompt for a time value with cout and thin use cin >> to read in a time value for the t4 variable.

Enter 17:44 during execution to verify it works

Part b

Output the value of t3 on its own line using cout <<

Part c

Output the value of t4 on its own line using cout <<

Suggested: compile and run the program before moving to the next task

Task 7 (17 points)

Part a

Implement the “greater than” operator >
Use the existing code for the “less than” operator as an example

Part b

Using your newly created greater than operator
Use an if-else statement to see if t1 > t2
Output a statement indicating the result
(e.g. “t1 is greater than t2\n” or “t1 is NOT greater than t2\n”)

Suggested: compile and run the program before moving to the next task

BONUS (5 pts extra)

Find and remove the 1 “obviously inappropriate” comment in the source code. You WILL be certain you have found it when you find it.

Prep for submit (1 point – correct name & compress)

In Ubuntu Linux browse to your A01 folder
Make sure your modified files are in it
Right click on the A01 folder, select compress
Check that things are set to tar.gz
Add your last name to the Filename, so it reads:
A01_YourLastName.tar.gz
Press the create button

Example: Pretend your last name is Golly
The compressed file would then be
A01_Golly.tar.gz

Start here

Browse to A01 folder

Right click on the A01 folder

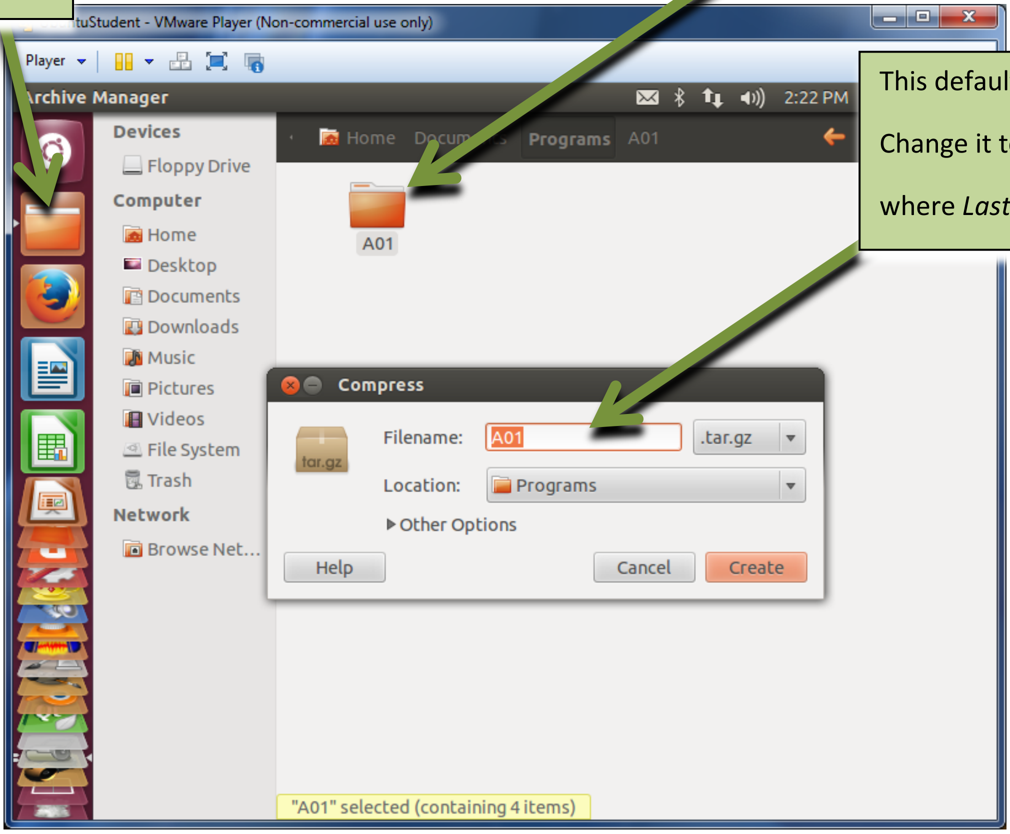
Select Compress

The pop-up shown will appear.

This defaults to say A01

Change it to read A01_LastName

where *LastName* is your last name



Filename: A01 .tar.gz
Location: Programs

Submit

Submit the **A01_LastName.tar.gz** file
to the correct course drop box in D2L

The drop box in D2L *should* be setup to overwrite submissions
Even if it allows multiple submissions, only the last submission will be graded
And the last submission's time stamp will be used to determine if the assignment was completed on time

Grading

A total of 100 points is possible.

Estimated points for each task are given next to the task headings
(which barring typing errors will sum to 100)

Bonus points may increase the score only to its maximum value
(i.e. 99 + 5 bonus = 100 total, 12 + 5 bonus = 17 total, etc)

Note the filename of what you turn in and correctly compressing the files is worth 1 point.
But if not done correctly may lead to there being nothing to grade

Note the submission task is not directly worth any points, but if not done successfully there will be nothing to grade

Note: Directions in this document take priority over those in the provided starting source code

Points will be awarded based on

comments, readability, formatting, and correctness of code
how well the directions in each of the above tasks were followed
compilability of the program (does it compile if not, why not)
executability of the program (does it run)
output of the program
time of submission
other items may also be considered at the discretion of the instructor

See the syllabus for other general grading policies.

You may consult with other students on how to accomplish the above tasks, but…

DO NOT COPY SOMEONE ELSE’s WORK

DO NOT LET SOMEONE ELSE COPY YOUR WORK