Matt Carleton 1/15 1:42 PM

yo Juni can i get your help on a simple c++ question when you get a minute

1/15 3:52 PM

Hi Matt, sure

Matt Carleton 1/15 4:06 PM

we're converting from mm to feet and inches

then have to return it as a string

this is the current code

```cpp
std::string convertToFeetAndInches(float length, float inOneInch)
{
        string s;
        int inches = abs(round(length / inOneInch));
        int feet = inches / inchesInFoot;
        if (feet > 0) {
                inches -= (feet * inchesInFoot);
                s = to_string(feet) + "' ";
        }
        s += to_string(inches) + '"';

        return s;
}
```

but inches needs to be a float cause we can have something like 3.5 inches

problem is when you to_string a float you end up with 3.50000

1/15 4:08 PM

ok so the 0000 should not be there?

Matt Carleton 1/15 4:08 PM

yeah this prints to a label so we dont really want all that trailing zeros

seems like a stringstream can do what i need

```cpp
string convertToFeetAndInches(float length, float inOneInch)
{
        int feet = length / (25.4*12);
        float inches = fmodf(length, 25.4*12) / 25.4;
        inches = (round(inches*100))/100;

        stringstream s;
        if(feet)
                s << feet << "\" ";
        s << inches << "\"";
        return s.str();
}
```

seems like a stringstream can do what i need

```
string convertToFeetAndInches(float length, float inOneInch)
{
        int feet = length / (25.4*12);
        float inches = fmodf(length, 25.4*12) / 25.4;
        inches = (round(inches*100))/100;

        stringstream s;
        if(feet)
                s << feet << "\" ";
        s << inches << "\"";
        return s.str();
}
```

but not sure how good these are to use

stringstream is fine, is even the right way

Matt Carleton 1/15 4:11 PM

are they much slower / heavier on the processor?

it should be fine in this cause cause its not called very often but i assume we dont want them in an execute loop or anything

1/15 4:11 PM

or maybe something like:
std::ostringstream oss;
oss.precision(1);
oss << std::fixed << inches;

ostringstream is good for that purpose, especially with output string

Matt Carleton 1/15 4:14 PM

oh yeah

i looked into the setprecision

1/15 4:14 PM

okay

Matt Carleton 1/15 4:14 PM

seems like that sets significant figures instead of decimal places

so 2.541 will go to 2.54

while 20.541 will go to 20.5

(with setprecision(3))

Matt Carleton 1/20 3:58 PM

hey Juni, just fixing some compiler warnings - what do you think is the best way to fix this

```
/* cycle through regions and find biggest gap */
for (int curr = 0, next = 1; next < regions.size(); curr++, next++) {
        float gap = regions[curr].gapBetween(regions[next]);
        if (gap > bestGap) {
                bestIndex = curr;
                bestGap = gap;
        }
}
```

```
MachineControl/Inkjet_Printing/CustomInkjet.cpp: In member function 'void CCustomInkjet::largestClearIOSpace(CCustomStick*, float*, float*)':
MachineControl/Inkjet_Printing/CustomInkjet.cpp:312:44: warning: comparison of integer expressions of different signedness: 'int' and 'std::vector<CToolingRegion>::size
_type' (aka 'long unsigned int') [-Wsign-compare]
  312 |         for (auto curr = 0, next = 1; next < regions.size(); curr++, next++) {
      |                                       ~~~~~^~~~~~~~~~~~~~~~~
```

1/20 3:59 PM

int vs. size_type?

Matt Carleton 1/20 3:59 PM

yeah

i can static cast the size_type to an int

but is that a little risky going from unsigned to signed

1/20 4:00 PM

yeah the one allows negative number (int) the other not (starting from 0 upwards). I prefer to cast the int to unsigned

you can easily define

```C++
    for (unsigned long int curr = 0, next = 0 ...
```

Matt Carleton 1/20 4:04 PM

oh yeah, that will still work fine when referencing an index in a vector right?

```
regions[curr].ga
```

1/20 4:05 PM

Yes, actually any unsigned data typ[e should be fine. The vector size should be (always) less than MAX_UINT right? not billions something

Matt Carleton 1/20 4:06 PM

yeah if the vector size was larger im not even sure what would happen to the vector.size() call

in this case we'll definitely be fine

I guess in LInux, size_type is an alias for unsigned long int (which is the largest what a non floating point can hold)

Matt Carleton 1/20 4:07 PM

MC

yep i think thats the right way to go thanks

```
/* cycle through regions and find biggest gap */
for (unsigned long curr = 0, next = 1; next < regions.size();
    curr++, next++) {
        float gap = regions[curr].gapBetween(regions[next]);
        if (gap > bestGap) {
                bestIndex = curr;
                bestGap = gap;
        }
}
```

1/20 4:08 PM

approved

👍

January 22

Matt Carleton 1/22 9:40 AM

MC

Hey Juni, are you able to review the fixed compiler warnings merge request on git?

its only a few lines so should be super easy

1/22 9:40 AM

Yes, I'll

yeah, you may merge it

Matt Carleton 1/22 9:43 AM

MC

sweet thanks

👍

February 3

Matt Carleton 2/3 1:33 PM

MC

hey Juni do you know much about bash scripting?

2/3 1:52 PM

Yeah, so so, what is the question?

Matt Carleton 2/3 1:57 PM

MC

do you know how i would run factory in a script?

even just running a single process like logger

im trying something like this

im trying something like this

```
LOGGER_DIR="/home/mattc/sysdev/factory2/Logger/"
LOGGER_CMD="./logger"
```

```
launch_logger(){
    echo "launching logger"
    cd $LOGGER_DIR
    echo $(pwd)
    . $LOGGER_CMD
    find_pid $LOGGER_BIN
    LOGGER_PID=$_RET
}
```

seems to change directory fine but all falls apart when i try run logger

```
[mattc@mattlaptop factoryTest]$ ./runScripts.sh
launching logger
/home/mattc/sysdev/factory2/Logger
./runScripts.sh: line 38: .: ./logger: cannot execute binary file
```

2/3 2:02 PM

remove the dot in launch_logger()?

Should be only:
$LOGGER_CMD
before find_pid

Matt Carleton 2/3 2:04 PM

ah yeap perfect

👍

and do you know if theres a way i can run all 4 processes?

(logger ldaemon machinecontrol & GUI)

should i be trying to open 4 terminals

or sending the processes to background

2/3 2:08 PM

put sufix & after each command like:

echo"Launching all processes..."
cd /home/mattc/sysdev/factory2
./Logger/logger &
./LicenseDaemon/ldaemon &
./MachineControl/machinecontrol &

```
cd /home/matt/sysdev/factory2
./Logger/logger &
./LicenseDaemon/ldaemon &
./MachineControl/machinecontrol &
./RollFormerGUI/RollformerQtGUI &.  # forget the binary name
```

Matt Carleton 2/3 2:08 PM

oh yeah so thats to run them in bg right?

2/3 2:09 PM

to kill, just use pkill <process or application name> so you don't need to store/save the PID

Matt Carleton 2/3 2:09 PM

yeah saving the PID is just so we can check that it didnt crash

2/3 2:09 PM

pkill, not sure if it's available in all distro or 32bit

ok2

Matt Carleton 2/3 2:10 PM

ideally we boot up all 4 processes then make sure all PID's are valid

2/3 2:10 PM

> Matt Carleton 2/3/2025, 2:08 PM
> oh yeah so thats to run them in bg right?

Yes

Matt Carleton 2/3 2:10 PM

meaning none of the processes crashed

awesome thanks for the help

2/3 2:10 PM

Anytime! 🥴

❤️

If it works, make it a nice script, I made similar but too much hardcoded. The scripts are start_cs.sh and kill_cs.sh in factory root folder.

Matt Carleton 2/3 2:13 PM

sweet, im basing it off run.sh for the moment

im looking at making it automatically go through each gmdl and just ensure no process crashes

2/3 2:18 PM

💯