KEEP IN MIND THAT I'M SELF-Taught, SO MY CODE MAY BE A LITTLE NOSEY. I'M SURE IT'S FAKE.

...Wow, this is like being in a house built by a child using nothing but a hatchet and a picture of a house.

IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE_APP CORRECT THAT ONLY KNOWS EXCEL FORMULAS.

IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EXITS UNTIL IT COMPILLED WITHOUT ERRORS. ONLY I UNDERSTOOD THE "I'M A SAFETY GUY."" Formulas...

THAT'S THE UGLIEST MESS OF CODE I'VE EVER SEEN. WHAT DO YOU WORK ON? ARE YOU WORKING ON?

IT'S NOT WORKING ITSELF, I SHUFFLE IT JUST LOOKS BAD BECAUSE IT'S A Spreadsheet Formula...

...WHICH ASSEMBLES A HAMMEL FUNCTION.

...FOR PARSONS XML...

...OH MY GOD...

IT'S OK! NOTHING DEPENDS ON THIS THAT WILL GIVE UP. LEAPING. DOES THAT MEAN WE CAN JUST THROW HAMMERS AT IT?

...I MEAN...

...WHAT, ANY?

...LIKE, YOU KNOW. ON A PHOTO OF A SCABBLE BOARD FROM A GAME WHERE THE BOARD RESIDENCE COULD COUNTED FOR THREE POINTS.

IT LOOKS LIKE SOMEONE TRANSFORMED A MARVEL SUPERHERO INTO A SCABBLE Board. I'M GLAD THEY MADE IT. REAL SWEEP ASL.
**THE INSPECTION PROCESS**

The inspection process follows any development operation whose product must be verified. As shown below, it consists of six operations, each with a specific objective:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLANNING</td>
<td>Materials to be inspected must meet inspection entry criteria.</td>
</tr>
<tr>
<td></td>
<td>Arrange the availability of the right participants.</td>
</tr>
<tr>
<td></td>
<td>Arrange suitable meeting place and time.</td>
</tr>
<tr>
<td>OVERVIEW</td>
<td>Group education of participants in what is to be inspected.</td>
</tr>
<tr>
<td></td>
<td>Assign inspection roles to participants.</td>
</tr>
<tr>
<td>PREPARATION</td>
<td>Participants learn the material and prepare to fulfill their assigned roles.</td>
</tr>
<tr>
<td>INSPECTION</td>
<td><em>Find defects.</em> (Solution hunting and discussion of design alternatives is discouraged.)</td>
</tr>
<tr>
<td>REWORK</td>
<td>The author reworks all defects.</td>
</tr>
<tr>
<td>FOLLOW-UP</td>
<td>Verification by the inspection moderator or the entire inspection team to assure that all fixes are effective and that no secondary defects have been introduced.</td>
</tr>
</tbody>
</table>
inspected. A clear statement of the project rules and changes to these rules along with faithful adherence to the rules go a long way toward practicing the required project discipline.

A prerequisite of process management is a clearly defined series of operations in the process (Figure 1). The miniprocess within each operation must also be clearly described for closer management. A clear statement of the criteria that must be satisfied to exit each operation is mandatory. This statement and accurate data collection, with the data clearly tied to trackable units of the process, are essential constituents of the information required for process management. In order to move the form of process management from qualitative to more quantitative, process terms must be more specific, data collected must be appropriate, and the limits of accuracy of the data must be known. The effect is to provide more precise information in the correct process context for decision making by the process manager.

In this paper, we first describe the programming process and places at which inspections are important. Then we discuss factors that affect productivity and the operations involved with inspections. Finally, we compare inspections and walk-throughs on process control.

[Fagan, Michael E. Design and code inspections to reduce errors in program development. IBM Systems Journal 15, 3 (September 1976), 182-211.]
**Code-Durchsicht ::**
Formalisierter Arbeitsschritt im Softwareentwicklungsprozess zur Aufdeckung von Defekten in bereits geschriebenen Quelltext.

**Softwaredefekt ::**
"A defect is an instance in which a requirement is not satisfied." (Fagan)

fehlende Übereinstimmung mit Spezifikation

Weiterführende Begriffe (variierender Formalisationsgrad):
formal technical review; (code) inspection (IEEE 1028); walk-through
Formalisierungsgrade variieren in der Praxis. Was folgt sind Optionen.
Spezifikation → Quelltext → Änderungen, Changelog (rework) → "Okay!" (follow-up) ("und keine neuen Fehler") → "Okay!" (follow-up) ("und keine neuen Fehler") → Änderungsvorschläge → Liste mit Defekten (inspection) → preparation → overview
Let's look if it's actually invalid... Probably not because:

http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap03.html#tag_03_276

So in my humble opinion, it's a valid name and thus systemd has a bug here.

Even if you assume leading numbers are not allowed, systemd running the unit under root is a bug too. It should refuse to run the unit, in particular in the light of:

systemd will validate all configuration data you drop at it

-- @poettering #6237 (comment)

This is clearly not the case. User=oday is clearly not properly validated and silently ignored and it's still a bug (or it's just put into something like atoi and will try to execute oday as uid 1, not sure about that).

So either way, there is a bug here.

In case of bug report: Unexpected behaviour you saw

The process started by systemd was root privilege

In case of bug report: Expected behaviour you didn't see

The process started by systemd should be user privilege

I wonder which tool permitted you to create it in the first place. Note that not permitting numeric UID to avoid ambiguities between numeric UID and textual user names.

So, yeah, I don't think there's anything to fix in systemd here. Understand this is annoying, but still: the username is clearly not valid.
Fehlerkultur

Kritik muss konstruktiv und höflich sein …

… und dankend angenommen werden.
**Figure 2** A study of coding productivity

- **NET CODING PRODUCTIVITY**
  
  \[ l_1 + l_2 + x = 123\% \]
  
  \[ l_1 + x + x = 112\% \]
  
  \[ x + x + x = 100\% \]

  SAMPLE SHOWED 23\% NET INCREASE

  122\% IN POST STUDY SAMPLE FROM NORMAL PRODUCTION (TO NORMALIZE FOR HAWTHORNE EFFECT)

- **NET SAVINGS (PROGRAMMER HOURS/K)** DUE TO:
  
  \[ l_1: 94, l_2: 51, l_3: -20 \]

- **REWORK (PROGRAMMER/HOURS/K. LOC)** FROM:
  
  \[ l_1: 78, l_2: 36, l_3: \]

- **QUALITY**

  AN INSPECTION SAMPLE HAD 38\% FEWER ERRORS/K. LOC THAN A WALK-THROUGH SAMPLE DURING EQUIVALENT TESTING BETWEEN POST UNIT TEST AND SYSTEM TEST IN THIS STUDY.

**Arbeitsweise der Gutachter**
Ziel ist: Defekte finden.

Oft: Zurückgreifen auf Wissen um typische Fehler.

Ändern sich mit der Zeit (Sprachen, Werkzeuge, Prozesse, Modultests, …)

Code-Completion, Linter, IDEs und z.B. Pair-Programming dienen als kontinuierliches Code-Review

Figure 5  Examples of what to examine when looking for errors at I,

1. Logic
   Missing
   1. Are All Constants Defined?
   2. Are All Unique Values Explicitly Tested on Input Parameters?
   3. Are Values Stored after They Are Calculated?
   4. Are All Defaults Checked Explicitly Tested on Input Parameters?
   5. If Character Strings Are Created Are They Complete, Are All Delimiters Shown?
   6. If a Keyword Has Many Unique Values, Are They All Checked?
   7. If a Queue Is Being Manipulated, Can the Execution Be Interrupted; If So, Is Queue Protected by a Locking Structure; Can Queue Be Destroyed Over an Interrupt?
   8. Are Registers Being Restored on Exits?
   9. In Queuing/Dequeuing Should Any Value Be Decremented/Incremented?
   10. Are All Keywords Tested in Macro?
   11. Are All Keyword Related Parameters Tested in Service Routine?
   12. Are Queues Being Held in Isolation So That Subsequent Interrupting Requestors Are Receiving Spurious Returns Regarding the Held Queue?
   13. Should any Registers Be Saved on Entry?
   14. Are All Increment Counts Properly Initialized (0 or 1)?

   Wrong
   1. Are Absolutes Shown Where There Should Be Symbolics?
   2. On Comparison of Two Bytes, Should All Bits Be Compared?
   3. On Built Data Strings, Should They Be Character or Hex?
   4. Are Internal Variables Unique or Confusing If Concatenated?

   Extra
   1. Are All Blocks Shown in Design Necessary or Are They Extraneous?

[Fagan, Michael, 1976]
– Lässt sich der Code übersetzen? Besteht er die definierten Testfälle?
– Hat der Gutachter den Code verstanden?
– Folgt der Code den etablierten Codekonventionen?
– Ist er verständlich kommentiert?
– Wird die Spezifikation erfüllt? Auch in Randfällen?
– Trifft der Code nicht-verallgemeinerbare Annahmen?
  (z.B. Verwendung von Pfadnamen auf der Autorenmaschine)
– Werden Fehlerfälle korrekt berücksichtigt?
– Gibt es redundanten/unverständlichen Code?