72-hours HOME EXAM – Resit exam august 2025
PG3401 C Programming

Permitted aids: All (except code generating AI)
Duration: 72 hours
Grading scale/evaluation type: National grading scale A - F
Date: August 5th – August 8th, 2025

The exam text is in English only, you can hand in your exam in either English or Norwegian (Swedish and Danish is also accepted). Programming language used should though be the language C-89, other languages than C will not earn points.

The exam text has 8 pages. There is a total of 4 tasks in the exam set.

There is a 72-hour deadline for this home exam. The period may overlap with other exams you have, so it is important that you use the time efficiently and plan your time well, so that you do not sit right before the deadline and have to submit several exams at the same time. Please note that the exam MUST be submitted within the deadline set in Wiseflow, and the assignment can only be submitted via WISEFLOW. It will not be possible to submit the assignment after the deadline – this means that you should submit well in advance so that you can contact the exam office or support if you have technical problems.

It is emphasized that the student must answer the exam independently and individually; cooperation between students and plagiarism is not allowed, the answer should represent the student's own work. This also means that it is not allowed to use code- or text-generating tools based on artificial intelligence. Be aware that attempts at "bluffing" on practical assignments in the form of screenshots found on the internet, screenshots received by other students or edited screenshots give negative points on the total score on the exam (which may result in the final grade being reduced by one or more grades), and in serious cases it will result in the filing of a cheating/plagiarism case. You should only document what you have achieved yourself, not try to make it look like you have achieved something you have not actually achieved. The exam must be solved on Linux.

Note that the tasks are made with increasing difficulty, and especially the last task is more difficult than the first tasks. Also note that because the resit exam has shorter time than the ordinary exam, the "easy" tasks are taken out of this resit exam, this is done to prevent using time on tasks all students are expected to solve regardless of skill-level, this means that the perceived difficulty of this resit exam might be higher, but the scale for the grades has been modified accordingly. Try to solve all tasks to the best of your ability, and document problems you have encountered well.

Format of submission

This is a practical programming exam (apart from task 1), so the focus should be on explaining how you have proceeded to solve the tasks, justifying your choices and presenting any assumptions you have made in your solution.

If you are unable to solve a task, it is better if you submit what you have done (even if it does not work), and explain how you have tried to solve problems you faced and what you did not achieve – than not answer the task at all. It is expected that everything works unless otherwise described in the PDF file, code that the student knows does not work should also be commented as such in the code. If you know that the program is crashing, not compiling or not working as intended, it is important to explain this along with what steps you have taken to try to solve the problem.

The submitted source code must be in 1 ZIP file, the name of the file must be PG3401_R25_[candidate number].zip. This file should have the following structure:

\ logdata \ ewa_log.txt
\ logdata \ ewa_data.bin
\ task2_[candidate number] \ makefile
\ task2_[candidate number] \ […]
\ task3_[candidate number] \ makefile
\ task3_[candidate number] \ […]
\ task4_[candidate number] \ makefile
\ task4_[candidate number] \ […]

In Wiseflow, you should upload a text answer with the name "PG3401_R25_[candidate number].pdf", and the ZIP file should be uploaded as an attachment to the text answer.

Be sure all files are included in the ZIP file (this includes all executable files, all makefiles, as well as the contents of the logdata folder). Each task-folder should have a makefile file, and no changes, third-party components or parameters should be required - the assessor will in shell on Debian Linux 10 go into the folder and type "make" and this should build the program with GCC. On all programming tasks, it is an **absolute requirement** to use the makefile file used during this course – as showed on **Lecture 4, slide 65 with the title «Use this makefile»**, you can make minor changes to the makefile file if necessary to solve the task, however, tasks that use a different makefile, regardless of whether it is auto-generated or created manually, will result in no points being awarded on the task. The referred lecture is available at the following URL: http://www.eastwill.no/pg3401/pg3401_lecture04.pdf

The text answer should contain an answer to assignment 1 (write it to the point, does not need a long thesis – however it should still be an answer of academic quality and style as expected of all work at college level). After the plain text answer, there should be 1 page of justification / documentation for each assignment, each of the documentation segments should be on a new page to make the text answer clear to the assessor (more than 1 page is allowed if the student deems it necessary,

readable screenshots are more important than keeping just 1 page). The answer must be in PDF format and have the correct file extension to be opened on both a Linux and a Windows machine (.pdf). Answers in other formats will not be read. (The text answer in the PDF file can be written on Mac or Windows, or any other operating system the student may be using, installing word processor on Linux is not required for this course.)

Required preparation

Before you start solving the programming tasks you need to download the tool EWA; this is new for this year's exam and has been covered during LECTURE 9 (streamed from Oslo March 20th) and demoed in the workshop held in Bergen April 3rd.

There is a new version for this resit exam, regardless if you downloaded the EWA tool during class or during the ordinary exam – you now have to download a new version.

You can either download the application from the domain 'eastwill.no' or download it from Canvas where it is found on the PG3401 coursepage under EKSAMEN (EXAM).

The student MUST use Debian 10 64 bit compiled for Intel (for students using VmWare Workstation on Windows or other Intel based systems) or compiled for ARM (for students using VmWare Fusion on Mac using M1, M2, M3 or M4 systems). No other versions of the application will be provided during the exam as only these 2 systems are supported.

To download from 'eastwill.no' follow these steps, the file must be downloaded, then unpacked, and execute privileges must be set manually. Note that it MUST be run from a location on your system where you have write-access or it will fail (no help will be given during the exam, understanding how to do this is required knowledge).

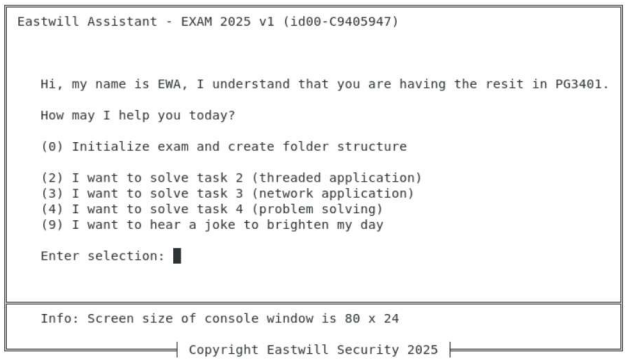
ZIP file is also found under EKSAMEN on Canvas, it can be copied in manually (from a USB drive for instance) if you don't have internet access in your VM. Access to the internet is not required for any task – but a functioning network adapter that can communicate on localhost (127.0.0.1) is required for Task 3 and 4 (network tasks).

VmWare Workstation (Intel based) or x64 Fusion (Mac with Intel CPU):

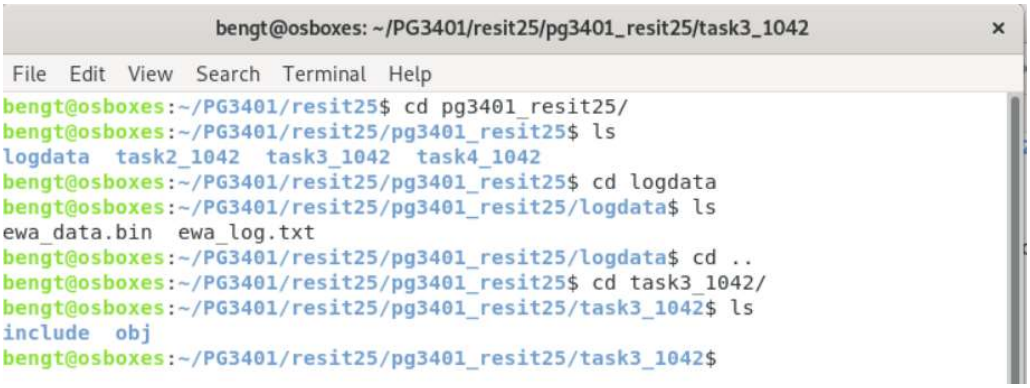
```
wget http://www.eastwill.no/pg3401/exam25/ewaexamv1_resit25_intel64.zip
unzip ewaexamv1_resit25_intel64.zip
chmod +x ewa_exam_r25_i64
./ewa_exam_r25_i64
```

VmWare Fusion (ARM based):

```
wget http://www.eastwill.no/pg3401/exam25/ewaexamv1_resit25_arm64.zip
unzip ewaexamv1_resit25_arm64.zip
chmod +x ewa_exam_r25_a64
./ewa_exam_r25_a64
```



Select option 0 to initialize the exam, when prompted enter your candidate number on this exam. This will create the folder structure you will use on this exam which should look like this:



Task 1. Theory (5 %)

a) Explain what the C programming language can be used for.

b) Who is Richard Stallman and what is he known for in the field of Information Technology?

c) What does the sudo command do on a Linux system, and why is that often needed by both developers and sysadmins?

Task 2. Threads (25 %)

In practical programming, it is often efficient to put time-consuming operations in worker-threads, examples of which are file operations, network operations and communication with external devices. In this assignment, you will simulate such operations with a smaller data set – in order to save you time during the exam, a ready-made application has already been created for you to change.

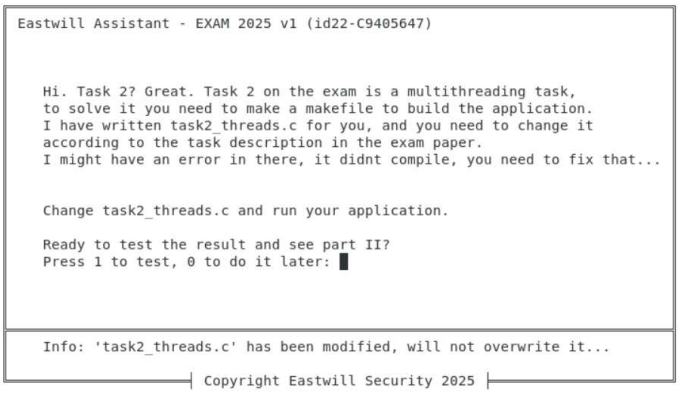
Application consists of 3 threads, the main thread (running the main function) and 2

worker threads. The main thread starts the worker threads with mechanisms for thread communication and synchronization (in this application, the main thread has no other function, and only starts the two threads that do the job themselves). The threads have a memory area with space of 4096 bytes for communication between the two threads.

One worker thread (thread A) read a text file, worker thread A should then send the file over to the other worker thread (thread B) through multiple cycles using the memory space described above, signaling thread B that there is data available in the buffer. Thread B then counts the number of instances of bytes with value 00, 01, 02, including; FF in the file it receives from thread A. Thread A and thread B both loop to process the file until it's finished. When the file is sent over in its entirety, worker thread A will end. Worker thread B completes its count of bytes, and then prints to the terminal window the number of instances of each of the 256 possible byte values, before it also exits. The main thread waits for both threads to exit, cleans up correctly, and then closes the application.

For this task you must first do the “Required preparation” as described above. In the EWA make sure you have chosen 0 to create and initialize the folder structure, and have supplied your candidate number for this exam.

In the EWA main interface select 2. This will create 2 files that you will use for this task; a source file with the code described above, and a txt file to be used for testing.



/pg3401_resit25/task2_[candidate number]/task2_threads.c
/pg3401_resit25/task2_[candidate number]/task2_pg2265.txt

The source file task2_threads.c is a solution to the application as was described above. The test file task2_pg2265.txt is the file you will use to test your multithreaded application (Hamlet by Shakespeare, taken from Project Gutenberg – originated from https://www.gutenberg.org/ebooks/2265).

PART I

The following changes must be done in task2_threads.c, do NOT create new files, and do NOT move the file to other folders (the EWA tool will read the file when you are ready, and expects it to be at the same location/file as it created it):

- You must create a makefile file (as detailed above under “Format of

submission") to build the program, make any changes required to the makefile file and source file to make the program build and run correctly
- The program uses global variables, change this to that all variables are local variables in the main function and submitted as parameter to the two threads
- Thread A has hardcoded the name of the file to be read, change the program so it takes the file name as a parameter on the command line and passes this on to Thread A as a variable
- Rewrite the code from using conditions to using semaphores
- Add code for explicit initialization of mutex and semaphores (with the *_init functions)
- Add comments to your code to document what the code does

PART II

- If you are able to complete the first part of the task, the tool EWA will prompt you for two (2) more changes you will make to your final solution.
- You need to (at least) partially succeed on Part I to be shown Part II in the EWA tool, this is required because Part II builds upon Part I of this task
- Once you have succeeded sufficiently the EWA tool will give you the next part of this task – your progress is shown in the tool

Required documentation:

You must take a screenshot of the EWA tool after you have completed part I (if you were able to do so). Your final ZIP file must include all files, including any output files from Part II (if you were able to complete Part II).

Task 3. Network (35 %)

For this task you must first do the "Required preparation" as described above. In the EWA make sure you have chosen 0 to create and initialize the folder structure, and have supplied your candidate number for this exam.

In the EWA main interface select 3. This will create a header file:

/pg3401_resit25/task3_[candidate number]/ include / ewpdef.h

In this header file you will find some define values and data structures. You are going to create a server application (and the EWA tool has implemented a client that will be used for testing your server). In order to run this you need to open 2 terminal windows; one running your application, and one running EWA.

The server application should be executed with port number and a user-specified ID on the command line, such as "task3 -port 25 -id FtpTest". When started, your application should open the specified port for LISTEN, for this task it is enough to

bind to loopback on 127.0.0.1, and communication must use TCP.

You should simulate traffic between client and server (the traffic is “inspired by FTP”), points will be given for implementations that are flexible and robust, with error handling that can correctly handle data received from the client (especially if it is wrong – or malicious).

Communicating with EWA involves the following steps:

- EWA will establish a TCP connection with your application (the server), EWA will attempt to connect to 127.0.0.1 and the port you provide to EWA
- Your server must send a “server accept” message to the client
- After receiving a LOGIN from the client, the server must reply with a code 130 "hello" message that includes the logged-in users name, and gives a greeting message
- After receiving the STORE command from the client, the server must reply with a 150 "server reply" message indicating that the server is ready to receive the file
- The file contents will then be received in a "client store data" structure and should be saved on disk, the size of the file is dynamic so your server needs to handle dynamic length to receive the file, after receiving the file the server needs to reply to the client with a "server reply" message with the code 226 to indicate transfer is complete
- The client can then send either a "close command" message with the QUIT keyword, or send a new file by sending a new DATA command
- When QUIT is received the server application should send a 221 message, and can then shut down the server application

The student needs to inspect the network data received from the EWA tool, and any output EWA gives to the user interface to implement this correctly. Any output in the form of warnings or errors should be considered hints to how to comply correctly.

The task should not be solved using Curl or other third-party libraries and should not be based on launching other applications in the operating system - only the use of Sockets as we have learned in lecture 10 on Network will give points on the task.

Required documentation:

You must take a screenshot of the program when it is running, the screenshot must be attached to the answer in an image file named task3_scrnshot.png (in the same folder as your makefile) AND must also be inserted in a suitable place in the text answer (PDF file). The screenshot must include both your running application, as well as the EWA tool after you have received a file (if you were able to do so). Any files received by the EWA tool must also be included in your exam answer, the files should have the filename as was sent as part of the DATA command and be stored in the same folder as your makefile (pg3401_resit25/task3_[candidate number]).

Task 4. Problem solving (35 %)

This task is intended as a more difficult task than the previous, and it will combine different techniques and functions with the need for some problem solving on your part. This task might use code that is part of the other tasks, so solving this task might be more difficult if you have not already completed the previous tasks.

For this task you must first do the “Required preparation” as described above. In the EWA make sure you have chosen 0 to create and initialize the folder structure, and have supplied your candidate number for this exam.

You will create an application that works as a network client, the EWA tool will bind to a port on the loopback adapter 127.0.0.1, and will listen on a random port, the port is displayed in the EWA tool when the port is ready to send data.

Your application must accept the IP address and port number of the server as parameters from the terminal, for instance:

```
./task4 -server 127.0.0.1 -port 42420
```

When you connect to the EWA tool, first select option 6 in the main interface, and it will then send you a file. The protocol used is a custom implementation best described as "TCP over TCP"-protocol. You need to use the protocol as defined in:

/pg3401_resit25/task4_[candidate number]/ include / ewpdef.h

You then need to create the code that uses this protocol correctly so the resulting code is able to receive the file sent. Your code need to handle out of sequence packets, check crc code, and send ACK or NACK responses as required. (Since NACK is not defined for TCP we use an existing value for NACK, see the header file.)

Note that some problemsolving and debugging needs to be expected to successfully implement the communication with EWA, the EWA tool will log errors if unexpected data is received which should serve as "hints" to what you need to change to comply.

The file received is in binary format, and contains a BMP image representing a butterfly. Finally, your code must save the file to disk, it should be saved in the same folder as your makefile.

Required documentation:

You must take a screenshot of the program when it is running, the screenshot must be attached to the answer in an image file named task4_scrnshot.png (in the same folder as your makefile) AND must also be inserted in a suitable place in the text answer (PDF file). The screenshot should show both your application and EWA. The file received must also be included in the exam submission as task4_received.bmp.

+

End of task set