Abgabefrist Übungsaufgabe 5

Die Lösung muss abhängig von der Tafelübung abgegeben werden, an der ihr teilnehmt:

- **W1** – Übung U5 in KW26 (24.06. - 27.06.): Abgabe bis **Mittwoch, den 03.07.2024 23:59**
- **W2** – Übung U5 in KW27 (01.07. - 04.07.): Abgabe bis **Montag, den 08.07.2024 10:00**

In darauffolgenden Tafelübungen werden teilweise einzelne abgegebene Lösungen besprochen, teilweise auch ein Lösungsvorschlag aus dem Tutorenteam.

Allgemeine Hinweise zu den BS-Übungen

- Es ist **nicht** mehr möglich, Einzelabgaben im AsSESS zu tätigen. Falls ihr (statt in einer Dreiergruppe) zu zweit oder zu viert abgeben möchtet, klärt dies bitte **vorher** mit eurem Übungsleiter! Der Lösungsweg und die Programmierung sind gemeinsam zu erarbeiten.
- Die abgegebenen Antworten/Programme werden automatisch auf Ähnlichkeit mit anderen Abgaben überprüft. Wer beim Abschreiben[1] erwischt wird, verliert ohne weitere Vorwarnung die Möglichkeit zum Erwerb der Studienleistung in diesem Semester!
- Die Zusatzaufgaben sind ein Stück schwerer als die „normalen“ Aufgaben und geben zusätzliche Punkte.
- Die Aufgaben sind über AsSESS (`https://sys-sideshow.cs.tu-dortmund.de/ASSESS/`) abzugeben. Dort gibt **ein** Gruppenmitglied die erforderlichen Dateien ab und nennt dabei die anderen beteiligten Gruppenmitglieder. Matrikelnummer, Vor- und Nachname sind bei dem Abgabevorgang in die entsprechenden Eingabefelder des AsSESS und nicht in die Abgabedateien einzutragen. Namen und Anzahl der abzugebenden C-Quellcodedateien[2] variieren und stehen in der jeweiligen Aufgabenstellung; Theoriefragen sind grundsätzlich in der Datei `antworten.txt`[3] zu beantworten. Bis zum Abgabetermin kann eine Aufgabe beliebig oft abgegeben werden – es gilt die letzte, vor dem Abgabetermin vorgenommene Abgabe.
- Sowohl für die Programmieraufgabe als auch für die Theorieaufgaben stellen wir Ihnen Vorlagen bereit, die Sie zur Bearbeitung der Aufgaben verwenden sollen. Diese werden im Moodle als ZIP-Ordner (`vorgaben-A5.zip`) zum herunterladen zur Verfügung gestellt.
- Sobald eine Abgabe korrigiert wurde, kann das Ergebnis ebenfalls hier eingesehen werden.
- Ihre Programme müssen von `gcc` (also kein C++) mit der Option `-Wall` kompilierbar sein (als Referenzumgebung dienen die Poolrechner, wie empfehlen zusätzlich `-Werror`). Sollten Warnungen entstehen können ihnen dafür Punkte abgezogen werden. Programm die nicht übersetzt werden können werden nicht als Lösung akzeptiert und mit maximal einem Punkt bewertet.
- Nutzen sie in ihrem Quelltext eine sinnvolle Einrückung für Blöcke und vermeiden sie mehrere Anweisungen in einer Zeile. In Fällen von unleserlicher Formatierung können ihnen Punkte abgezogen werden.

[1] Da wir im Regelfall nicht unterscheiden können, wer von wem abgeschrieben hat, gilt das für Original **und** Plagiat.
[2] codiert in UTF-8
[3] reine Textdatei, codiert in UTF-8

Aufgabe 5: Dateioperationen (10 Punkte)

In dieser Aufgabe wird der Umgang mit dem Dateisystem geübt.

ACHTUNG: Zu dieser Aufgabe existiert eine Vorgabe in Form von C-Dateien mit einem vorimplementierten Code-Rumpf, die Sie in der Programmieraufgabe erweitern sollen. Diese Vorgaben sind von der Moodle-Seite herunterzuladen, zu entpacken und zu vervollständigen! Nutzen Sie auch für die Theorieaufgaben die Vorlage (`antworten.txt`).

5.1 Theorieaufgaben (4 Punkte)

1. Unix-Zugriff auf Peripheriegeräte (2 Punkte)

Welche zwei Klassen von Geräten gibt es (Die Bezeichnung reicht jeweils)? Beschreiben Sie in eigenen Worten den Unterschied zwischen diesen Klassen von Geräten und nennen Sie für jede Klasse ein Beispiel (Ein Beispiel reicht jeweils aus).

⇒ `antworten.txt`

2. E/A-Scheduling (2 Punkt)

Betrachten Sie einen Plattenspeicher mit 16 Spuren. Der E/A-Scheduler bekommt immer wieder Aufträge für eine bestimmte Spur. Die Leseaufträge in L_0 sind dem E/A-Scheduler bereits bekannt. Nach einem bearbeiteten Auftrag erhält er die Aufträge in L_1. Nach vier weiteren (d.h. nach insgesamt fünf) bearbeiteten Aufträgen erhält er die Aufträge in L_5. Zu Beginn befindet sich der Schreib-/Lesekopf über Spur 0. Die Aufträge lauten

$$L_0 = \{3, 7, 9, 15\}, L_1 = \{2, 13\}, L_5 = \{0, 7, 8, 12\}.$$

Tragen die Reihenfolge der gelesenen Spuren für einen E/A-Scheduler in Ihr `antworten.txt` ein, der nach der Fahrstuhlstrategie (Elevator) arbeitet.

⇒ `antworten.txt`

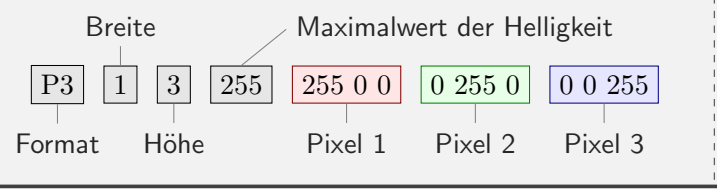
5.2 Programmieraufgabe: Portable Anymap (6 Punkte)

Für die Programmieraufgabe sollen Sie sich mit dem *Portable Anymap*-Dateiformat beschäftigen. Eine Erklärung des Formats finden Sie im Moodle:

`Tutorium ⇒ Weitere Übungsmaterialien ⇒ Anymap-Format`

-

beispiel1.ppm



Für die Bearbeitung dieser Aufgabe sind die Funktionen **fopen(3)**, **fclose(3)**, **fseek(3)**, **ftell(3)**, **fscanf(3)** und **fprintf(3)** hilfreich.

a) Format lesen (2 Punkte)

Ziel dieser Teilaufgabe ist es ein C-Programm (`a5_a.c`) zu entwickeln, das eine Datei im *Portable Anymap*-Format einlesen kann und die Breite und Höhe des enthaltenen Bildes, sowie die Dateigröße (in Bytes) in der Konsole ausgibt. Der Pfad für die zu lesende Datei soll als Kommandozeilenparameter beim Aufruf übergeben werden. Der Aufruf muss wie folgt aussehen:

./a5_a <Dateipfad>

Beispiel

```
studi@bsvm:~$ ./a5_a beispiel1.ppm
Breite: 1
Hoehe: 3
Dateigroesse: 34 bytes
```

In den Vorgaben zu dieser Aufgabe finden sich einige Beispieldateien, sowie eine Codevorgabe für den Ausgabestring.

Es sollen hier nur trivial formatierte (ASCII-kodierte) Dateien betrachtet werden, Sonderfälle wie Kommentarzeilen sind NICHT zu beachten.

Denken Sie an eine ordentliche Fehlerbehandlung, zum Beispiel bei falscher Nutzereingabe, und achten Sie darauf, dass Ihr Programm ordnungsgemäß hinter sich aufräumt.

`⇒ a5_a.c`

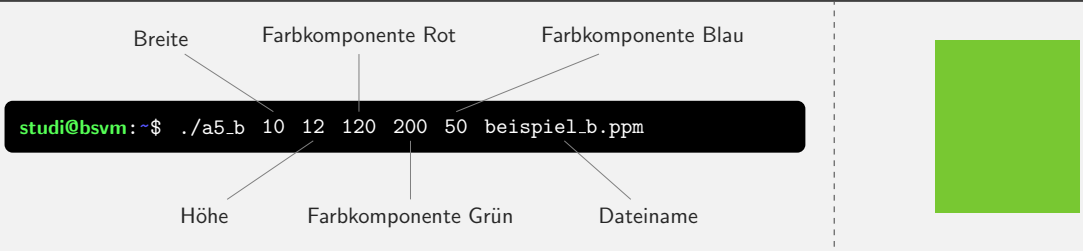
Hinweis: Es wird nur eine Vorlage für a) mitgeliefert. Kopieren sie diese für die weiteren Teilaufgaben und erweitern sie ihr Programm schrittweise.

b) Format schreiben (2 Punkte)

In dieser Teilaufgabe soll ein C-Programm (`a5_b.c`) implementiert werden, bei dem ihr eine Bilddatei im *Portable Pixmap*-Format mit einer Farbe durchgehend füllt. Hierbei sollen die Pixeldaten in ASCII kodiert werden, der Maximalwert für die Helligkeit soll 255 sein. Die Breite, Höhe und zu verwendende Farbe sollen als Kommandozeilenparameter beim Aufruf übergeben werden; zusätzlich soll noch der Pfad, in dem das Bild abgespeichert wird, angegeben werden. Der Aufruf muss wie folgt aussehen:

./a5_b <Breite> <Höhe> <Farbkomponente rot> <Farbkomponente grün> <Farbkomponente blau> <Dateiname>.ppm

Beispiel



Die Ausgabedatei `beispiel_b.ppm` finden Sie in der Vorlage.

Denken Sie an eine ordentliche Fehlerbehandlung, zum Beispiel bei falscher Nutzereingabe, und achten Sie darauf, dass Ihr Programm ordnungsgemäß hinter sich aufräumt.

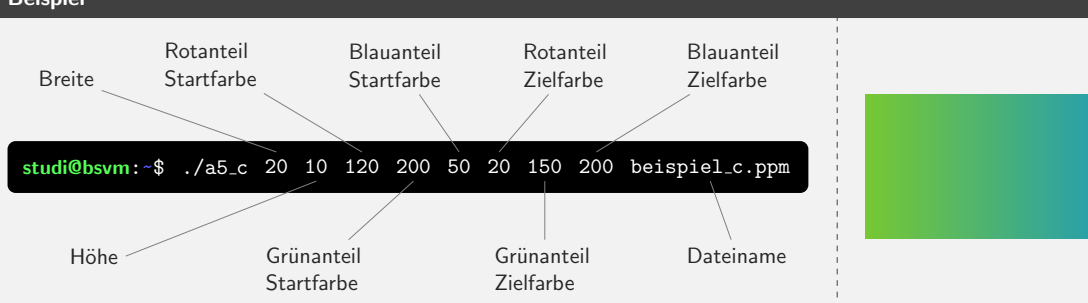
⇒ `a5_b.c`

c) Farbgradient (2 Punkte)

In dieser Teilaufgabe soll ein C-Programm (`a5_c.c`) implementiert werden, das einen Farbgradienten in einer Datei im *Portable Pixmap*-Format abspeichert. Hierbei sollen die Pixeldaten in ASCII kodiert werden, der Maximalwert für die Helligkeit soll 255 sein. In der Mitte des Gradienten sollen Start- und Zielfarbe im gleichen Verhältnis vorkommen. Der Gradient soll von einer Startfarbe gleichmäßig in eine Zielfarbe übergehen. Der Gradient soll von links nach rechts verlaufen. Hierbei sollen Breite und Höhe, sowie die Start- und Zielfarben als Kommandozeilenparameter beim Aufruf übergeben werden; zusätzlich soll noch der Pfad, in dem das Bild abgespeichert wird, angegeben werden. Der Aufruf muss wie folgt aussehen:

./a5_c <Breite> <Höhe> <Rotanteil Startfarbe> <Grünanteil Startfarbe> <Blauanteil Startfarbe> <Rotanteil Zielfarbe> <Grünanteil Zielfarbe> <Blauanteil Zielfarbe> <Dateiname>.ppm

Beispiel



Die Ausgabedatei `beispiel_c.ppm` finden Sie in der Vorlage.

Denken Sie an eine ordentliche Fehlerbehandlung, zum Beispiel bei falscher Nutzereingabe, und achten Sie darauf, dass Ihr Programm ordnungsgemäß hinter sich aufräumt.

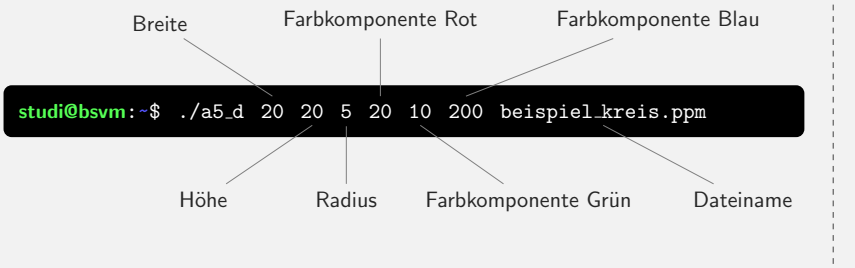
⇒ `a5_c.c`

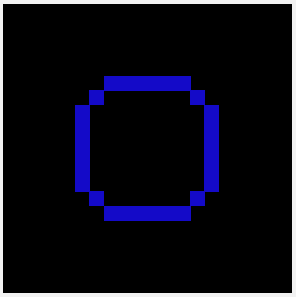
d) Zusatzaufgabe: Kreis zeichnen (2 Sonderpunkte)

Für die Zusatzaufgabe soll ein C-Programm (`a5_d.c`) erstellt werden, das ein Bild mit einem Kreis zeichnet und in einer Datei im *Portable Pixmap*-Format abspeichert. Die Breite und Höhe des Bildes, sowie der Radius des Kreises in Pixeln und dessen Farbe sollen als Kommandozeilenparameter beim Aufruf übergeben werden; zusätzlich soll noch der Pfad an dem das Bild abgespeichert wird angegeben werden. Der Aufruf muss wie folgt aussehen:

./a5_d <Breite> <Höhe> <Radius> <Farbkomponente rot> <Farbkomponente grün> <Farbkomponente blau> <Dateiname>.ppm

Beispiel





Die Ausgabedatei `beispiel_kreis.ppm` finden Sie in der Vorlage.

Der Kreis soll zentriert im Bild gezeichnet werden, die Mitte des Kreises ist also die Mitte des Bildes.

Denken Sie an eine ordentliche Fehlerbehandlung, zum Beispiel bei falscher Nutzereingabe, und achten Sie darauf, dass Ihr Programm ordnungsgemäß hinter sich aufräumt.

⇒ `a5_d.c`

Tipps zu den Programmieraufgaben:

- Kommentieren Sie den Quellcode ausführlich, so dass wir auch bei Programmierfehlern im Zweifelsfall noch Punkte vergeben können!
- Es ist unbedingt daran zu denken, dass viele Systemaufrufe fehlschlagen können! Diese Fehlerfälle sind abzufangen (die Aufrufe melden dies über bestimmte Rückgabewerte, siehe die jeweiligen man-Pages) und durch entsprechende Fehlermeldungen sichtbar zu machen (z.B. unter Zuhilfenahme von **perror(3)**). Das Programm ist danach ordnungsgemäß zu beenden.
- Die Programme sollen sich mit dem gcc auf den Linux-Rechnern im IRB-Pool übersetzen lassen. Es ist das mitgelieferte Makefile (Kommando `make`) zu verwenden, oder der Compiler mit den folgenden Parametern aufzurufen::
  `gcc -std=c11 -Wall -o ziel datei.c`
  Weitere (nicht zwingend zu verwendende) nützliche Compilerflags sind: `-Wpedantic -Werror -D_POSIX_SOURCE`
- Achten Sie darauf, dass sich der Programmcode ohne Warnungen übersetzen lässt; z.B. durch Nutzung von `-Werror`.