Portfolioprüfung - Werkstück A

Thema: Dynamische Partitionierung

Melina Friedrich

Das Ziel der vorliegenden Arbeit war es ein Simulationsprogramm, anhand verschiedener Realisierungskonzepte für das Speicherverwaltungskonzept der dynamischen Partitionierung zu entwickeln. Für die Lösung dieser Aufgabe sollte die Bash-Shell verwendet werden. Der Benutzer sollte hierbei anhand verschiedener Eingabeoptionen die Speicherbelegung steuern können und im Ergebnis eine visuelle Ausgabe der Speicherbelegung mit zugehörigen Informationen zu dieser erhalten. Diese Darstellung erfolgt in der Kommandozeile.

In der Dokumentation werde ich darauf eingehen, wie sich das Projekt entwickelt hat. Über erste Startschwierigkeiten berichten und wie die tatsächliche Realisierung schlussendlich erfolgte. Von der Ermittlung der größten und kleinsten angelegten Partition, sowie Ermittlung der freien oder belegten Blöcke bis hin zur Erzeugung der farblich visuellen Darstellung der Speicherbelegung.

Erste Schritte und Überlegungen

Wie vereinbart arbeitete ich bei meinem ersten Entwurf mit Testwerten und habe mir daraufhin eine Simulationsumgebung mit selbst angelegten Arrays erstellt. Dies funktionierte einwandfrei. Beim Versuch des Zusammenfügens der Teile stellte sich leider heraus, dass die Teile sich an den Schnittstellen nicht miteinander verbinden ließen. So wie die Arrays tatsächlich vorlagen, ließen sie sich nicht abgreifen. Aus diesem Grund musste ich ein neues Skript entwerfen. Dafür nahm ich mir das Skript bestfit zur Hilfe und suchte mir meine Werte, mit denen es sich arbeiten ließ (safe_name, safe_groesse, seekGroesse und speicherout). Hierfür hatte ich mir am 23.06 die Daten von GitHub heruntergeladen und mich daran orientiert. In der finalen Version vom 24.06 waren allerdings nicht mehr alle Werte enthalten. Daraufhin habe ich den fehlenden Programmteil in den Startbereich meiner Funktion eingebunden, da es sich um einen notwendigen Bestandteil für meinen Skript gehandelt hatte. Der Bestandteil der dann in der finalen Version entfernt wurde habe ich der Nachvollziehbarkeit in meinem Skript beibehalten. Dieser Abschnitt am Anfang meiner Funktion ist dort deutlich gekennzeichnet.

Zuallererst habe ich geschaut, wie ich aus den vorhandenen Informationen Werte ermitteln oder berechnen kann, die für die spätere Ausgabe oder im weiteren Programmverlauf genutzt werden könnten, um weitere Berechnungen vornehmen zu können. So wurden z.B. Daten für die Ermittlung der Gesamtspeichergröße aus save_speicherplatz entnommen und die tatsächliche Belegung aus dem Array save_speicher gezogen. Dies geschieht anhand einer for-Schleife. Da die Werte seekGroesse und summe bereits in der vorauslaufenden Funktion vorhanden sind, werden diese im nächsten Schritt für die weitere Berechnung des belegten Speichers übernommen. Danach wird anhand der vorliegenden Speicherbelegung und Speichergröße der Grad der Speicherbelegung in Prozent berechnet. Diese beiden Berechnungen werden dann zusätzlich später für die Bildschirmausgabe mit den Informationen zur Speicherbelegung verwendet.

Die Speicherbelegung wird an die Variable elements übergeben um eine übersichtliche Darstellung in den Schleifen zu gewährleisten, was gegebenenfalls auch Schreibfehler verhindert. Das Array speicherout enthält die aktuelle Belegung des Speichersystems und enthält die Informationen, welche der Speicherplätze nicht gefüllt sind

Im Array save_name wird jede Partition einmalig aufgelistet. Für nicht vorhandene Partitionen, die eine Lücke erzeugen wird eine Stelle im Array nicht besetzt. Diese Informationen übergebe ich an die Variable partitions.

Ermittlung der größten und kleinsten Partition

Danach wird die größte und kleinste Partition ermittelt. Zu Beginn wird die Variable max auf 0 gesetzt. Dann wird mit einer for-Schleife durch das Array save_groesse gegangen und anhand aller enthaltenen Elemente ermittelt wie groß die größte enthaltene Partition ist und das Ergebnis an die Variable max übergeben. Nach demselben Prinzip wird bei der Suche nach der kleinsten enthaltenen Partition vorgegangen; nur das hier die Bedingung nicht greater-than sondern less-than ist um dies zu ermitteln. Im ersten Anlauf habe ich versucht den ersten angelegten Mindestwert am Anfang des Arrays abzugreifen. In weiteren Testversuchen mit den unterschiedlichsten Belegungsvarianten ist das Programm in einen Fehler gelaufen, weil das Array an der Stelle nicht immer befüllt war, sodass

der tatsächlich vorhandene Mindestwert nicht ermittelt werden konnte. Daher habe ich den Startwert für die Mindestmenge auf den Wert “max int“ = 2147483647 gesetzt. Wird am Ende des Durchlaufens des Arrays festgestellt, dass kein kleiner Wert gefunden wurde enthält das Array keine Informationen zu einer vorhandenen Partition. Ist das der Fall wird dies in einer if-Abfrage abgeprüft und der Wert auf 0 gesetzt.

```
max=0
for pos in ${save_groesse[*]}
do
  if [ $pos -gt $max ]
  then
    max=$pos
  fi
done
```

Abbildung 1.1. Ermittlung der größten Partition.

```
min=2147483647
for pos in ${save_groesse[*]}
do
  if [ $pos -lt $min ]
  then
    min=$pos
  fi
done
# Wird keine Partition gefunden, wird der Wert auf "0" gesetzt.
if [ $min -eq 2147483647 ]; then min=0; fi
```

Abbildung 1.2. Ermittlung der kleinsten Partition.

Ermittlung der freien und belegte Blöcke

Für die spätere Ausgabe wird im nachfolgenden ermittelt wie viele Speichereinheiten belegt sind oder frei sind. Zu Beginn werden die Variablen anzahlBloecke und freieBloecke auf 0 gesetzt. Danach wird mit einer for-Schleife durch das Array speicherout gegangen. Im Array speicherout sind die einzelnen Speicherplätze, wenn sie belegt sind als Ziffern (int), oder wenn sie nicht belegt sind als X (char) dargestellt. Es wird dann abgefragt, ob das enthaltene Element ungleich X ist, wenn dies der Fall ist wird bei der Variablen anzahlBloecke jeweils um einen Wert hochgezählt. Wird ein X gefunden verzweigt das Programm in die else-Anweisung und zählt die Variable freieBloecke um eins hoch. Dies wird solange ausgeführt bis der komplette Speicherbereich durchlaufen wurde.

```
anzahlBloecke=0
freieBloecke=0
for (( i=0; i<$elements; i++ ))
do
  pos=${speicherout[$i]}
  if [ $pos != "X" ]
  then
    anzahlBloecke=$(expr $anzahlBloecke + 1 )
  else
    freieBloecke=$(expr $freieBloecke + 1)
  fi
done
```

Abbildung 2 Ermittlung der Anzahl an belegten und freien Blöcken

Externe Fragmentierung

Um die externe Fragmentierung berechnen zu können muss ermittelt werden, welcher der größte freie Block innerhalb der Speicherbelegung ist. In der Variable tempmax_fB wird ein gefundener Bereich hochgezählt. Wird beim Durchlaufen eine neue Partition erreicht und der Prüfwert char x wechselt in einen int Wert wird geprüft, ob die temporär ermittelte Größe größer dem bisher bekannten Wert ist. Ist dies der Fall wird der neue Höchstwert an die Variable max_fB übergeben. Für die fortlaufende Zählung weiterer gefundener freier Bereiche wird der Zähler wieder auf 0 gesetzt. Am Ende der Prüfung kann ein leerer block entstehen, der nochmal gegen die bisher ermittelte größte Lücke (max_fB) abgeglichen wird. Für die Ermittlung der Berechnung der externen

Fragmentierung habe ich mich einer Berechnungsfunktion des awk bedient.

```
max_fB=0                # Groesster freier Wert innerhalb
tempmax_fB=0    # Hier wird der Wert einer Luecke hochgeza
for (( i=0; i<$elements; i++ ))
        do
        pos=${speicherout[$i]}
        if [ $pos = "X" ]
        then
        tempmax_fB=$(expr $tempmax_fB + 1)
        fi
        if [ $pos != "X" ] && [ $tempmax_fB -ge $max_fB ]
        then
        max_fB=$tempmax_fB
        tempmax_fB=0
        fi
```

Abbildung 3.1. Ermittlung des größten freien Blockes.

```
if [ $tempmax_fB -ge $max_fB ]
then
groessterfreierBlock=$tempmax_fB
else
groessterfreierBlock=$max_fB
fi
```

Abbildung 3.2. Abgleich mit max_fB

Erzeugung der farblichen visuellen Darstellung der Speicherbelegung

Für die farbliche Darstellung des Speichersystems wird das Array farbe genutzt. Dort kann jeder belegte oder unbelegte Speicherplatz mit den dazugehörigen Farbinformationen abgespeichert werden. Im Array speicherout enthaltene Werte werden in einer if- else Abfrage gegen die ermittelten maximal($max), minimal($min) Werte verglichen. Je nachdem welche Bedingung erfüllt ist, wird diesem dann ein Farbwert zugeordnet. Diese Abfrage läuft in einer for-Schleife über den gesamten Speicherbereich. Jeder einzelne Speicherplatz wird mit einer Farbinformation in eine Arrayposition vom Array farbe gespeichert.
Zu Beginn, wenn das Speichersystem nicht gefüllt ist oder alle Partitionen gelöscht wurden, werden alle Speicherplätze weiß angezeigt. Wird eine erste Partition angelegt ist sie weder die Größte noch die Kleinste und wird daher in blau ausgegeben. Kommt eine zusätzliche Partition hinzu wird die Kleinere von beiden in grün angezeigt und die Größere in rot. Bei gleicher Größe werden Beide in blau angezeigt. Kommen weitere zusätzliche Partitionen hinzu werden alle die, die denselben größten Wert haben rot angezeigt. Genauso ist es bei den kleinsten; nur in grün. Alle weiteren Partitionen werden blau und freie Bereiche weiß angezeigt. Für die Farbausgabe werden die einzelnen Arrayposition fortlaufend ausgegeben. Die dabei enthaltene -e option im echo Ausgabebefehl bewirkt, dass das Parsen der Escape-Sequenzen in den Farbvariablen in der Bildschirmausgabe aktiviert wird. Unterhalb davon gibt es dort noch eine Legende zur Erklärung des Farbschemas.

```
for (( i=0; i<$elements; i++ ))
do
  pos=${speicherout[$i]}
  if [ $pos == "X" ]
  then
    farbe[$i]=$( echo -e $c1"+" )
  else
    if [ $pos -eq $max ] &&  [ $pos -ne $min ]
    then
      farbe[$i]=$( echo -e $c3"+" )
    fi
    if [ $pos -eq $min ] && [ $pos -ne $max ]
    then
      farbe[$i]=$( echo -e $c4"+" )
    fi
    if [ $pos -eq $min ] && [ $pos -eq $max ]
    then
      farbe[$i]=$( echo -e $c2"+" )
    fi
    if [ $pos -ne $max ] && [ $pos -ne $min ]
    then
      farbe[$i]=$( echo -e $c2"+" )
    fi
  fi
done
# Der komplette entstandene String wird dann dem
# Durch die Mitgabe der -e option im echo Ausgab
farbe[$elements]=$( echo -e $c0 )
```

Abbildung 4. Ermittlung der Farbbelegung für die visuelle Darstellung der Speicherbelegung

Am Ende werden Belegungskennzahlen in tabellarischer Form ausgegeben. Diese beinhalten z.B. Informationen zur Belegung und Fragmentierung.

Darunter beginnt die Ausgabe der Partitionen als Liste. Im Array save_name wird geprüft, ob dort ein Partitionsname angegeben ist. Wenn dies der Fall ist, wird dieser ausgegeben und im Array speicherout die dazugehörige Größe abgefragt und ausgeben.
Ganz zum Schluss werden nur noch die bereits vorbereiteten Berechnungen und Ermittlungen (wie z.B. der Grad der externen Fragmentierung) in einer Übersicht aufgerufen.

Resüme

Alles in allem zeigt sich, dass für eine Projektarbeit in diesem Umfang, das Gespräch zur Herangehensweise und Ausarbeitung der Schnittstellen den Grundstein für ein erfolgreiches Projekt legt. Der regelmäßige Austausch aller Mitglieder des Projektteams über den aktuellen Fortschritt ihrer Arbeit und gegebenenfalls neue Ansätze innerhalb ihres Projektteils sind für die Realisierung des Projekts essentiell. Darüber hinaus ist es zwingend erforderlich, dass sich an die getroffenen Absprachen gehalten wird oder die Teammitglieder zeitnah über Änderungen Informiert werden. Auch sollte während der Zusammenarbeit ein konstruktiver und sachlicher Umgang mit den anderen Beteiligten gewählt werden.
Positiv lässt sich feststellen, durch die Praxis in Shell werden Kenntnisse und Fertigkeiten vertieft, die sich auch in anderen Projekten wiederverwenden lassen. Nur durch die praktische Anwendung kann man Sicherheit mit der Programmierung in Shell Skripten entwickeln. Die Plattform GitHub hat sich als gutes Tool erwiesen um gemeinsam an einem Projekt zu arbeiten.

Quellenangaben

011 Fabien LOISON - www.flogisoft.com
Bash tips: Colors and formatting (ANSI/VT100 Control sequences)
https://misc.flogisoft.com/bash/tip_colors_and_formatting

Shell Scripting Tutorial
Functions
Function
https://www.shellscript.sh/functions.html

Tipps und Tricks
von Michael Luthardt
Rechnen mit und in der bash shell
https://dr-luthardt.de/linux.htm?tip=calc

ubuntu Deutschland e.V.
https://forum.ubuntuusers.de/topic/bash-script-dezimalzahlen-multiplizieren-und-r/
26. September 2014 10:04 – Autor: glaskugel

Unix & Linux Stack Exchange is a question and answer site for users of Linux, FreeBSD and other Un*x-like operating systems.

Adding two numbers using expr
https://unix.stackexchange.com/questions/44995/adding-two-numbers-using-expr

wikipedia.org
2147483647 - 32-Bit-Integer-Limit bei Computern
https://de.wikipedia.org/wiki/2147483647

Rheinwerk Computing / Openbook / Shell-Programmierung
Ganze Zahlen vergleichen - Zeichenketten vergleichen
https://openbook.rheinwerk-verlag.de/shell_programmierung/shell_006_003.htm