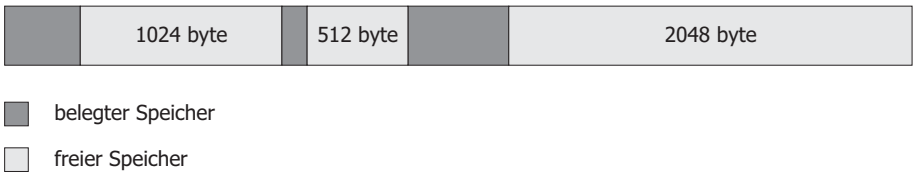
5. Übung

Ausgabe	Abgabe	Diskussion
15.11.13	22.11.13	25.11.-29.11.13

Bitte bei der Abgabe Name/Matr.Nr. der Mitglieder einer Gruppe, Nummer der Übung/Teilaufgabe und Datum auf den Lösungsblättern nicht vergessen! Darauf achten, dass die Lösungen beim richtigen Tutor abgegeben werden. Achten Sie bei Programmieraufgaben außerdem darauf, dass diese im Linuxpool kompilierbar sind.
Zu spät abgegebene Lösungen werden nicht berücksichtigt!

Aufgabe 1: Speicherverwaltung (2+2=4 Punkte)

Gegeben sei folgende Belegung eines Speichers:



Weiterhin seien vier Belegungensmethoden für Speicherplatzanforderungen gegeben:

First-Fit: Belege von vorne beginnend den ersten freien Speicherbereich, der groß genug ist, die Anforderung zu erfüllen.

Rotating-First-Fit: Wie First-Fit, jedoch wird von der Position der vorherigen Platzierung ausgehend ein passender Bereich gesucht. Wird das Ende des Speichers erreicht, so wird die Suche am Anfang des Speichers fortgesetzt (maximal bis zur Position der vorherigen Platzierung). Bei der ersten Anforderung beginnt die Suche am Anfang des Speichers.

Best-Fit: Belege den kleinsten freien Speicherbereich, in den die Anforderung passt.

Worst-Fit: Belege den größten freien Speicherbereich,in den die Anforderung passt.

a) Wie sieht der obige Speicher aus, wenn nacheinander vier Anforderungen der Größe 384 Byte, 640 Byte, 512 Byte und 2048 Byte ankommen? Notieren Sie für jede Strategie die freien Speicherbereiche nach jeder Anforderung (z.B. (1024, 512, 2048) für die obige Belegung), und geben Sie an, für welche Methoden die Anforderungen erfüllt werden können!

   Beachten Sie, dass die Daten jeweils linksbündig in einer Lücke abgelegt und einmal belegte Speicherbereiche nicht wieder freigegeben werden!

b) Sie dürfen nun die Reihenfolge der freien Speicherbereiche und die der Anforderungen vertauschen (aber keine Speicherbereiche zusammenfassen). Geben Sie für jede Strategie eine Speicherbelegung und die zugehörigen Anforderungen an, die für jeweils nur diese Methode erfolgreich arbeitet und für die übrigen Methoden nicht alle Anforderungen erfüllen kann. Geben Sie auch die Speicherbelegungen nach Abarbeitung der einzelnen Anforderungen an.

Aufgabe 2: Speicherverwaltung (2+3+1=6 Punkte)

In der Vorlesung haben Sie das Prinzip des *Pagings* kennengelernt. Es beinhaltet, dass im Hauptspeicher eine begrenzte Zahl von Speicherseiten (*Pages*) fester Größe gehalten wird, die kleiner als die Anzahl der tatsächlich existierenden Seiten ist. Seiten, die gerade nicht im Hauptspeicher gehalten werden können, werden auf ein externes Speichermedium ausgelagert.

Will ein Prozess auf eine solche Seite zugreifen, muss sie zuvor vom externen Speicher in den Hauptspeicher zurückgeholt werden (*Seitenfehler, Page Fault*). Sind noch nicht alle Plätze für Speicherseiten (*Frames*) im Hauptspeicher gefüllt, stellt dies kein Problem dar. Ist jedoch kein Platz mehr vorhanden, muss eine Seite im Hauptspeicher ausgewählt werden, die ausgelagert und durch die neue Seite ersetzt wird.

Für die Auswahl der zu ersetzenden Seite gibt es verschiedene Strategien:

FIFO (First In, First Out): Die jeweils älteste Seite wird aus dem Hauptspeicher ausgelagert.

LRU (Least Recently Used): Es wird die Seite aus dem Hauptspeicher entfernt, auf die am längsten kein Zugriff erfolgt ist.

SC (Second Chance): Diese Strategie ist eine Kombination aus FIFO und LRU. Pro Seite wird ein Use-Bit gespeichert, welches gesetzt wird, wenn nach dem Zugriff, der die Seite angefordert hat, ein weiterer Zugriff erfolgt. Ersetzt wird zuerst die älteste Seite mit nicht gesetztem Use-Bit. Bei einem Seitenfehler oder wenn die Use-Bits aller Seiten gesetzt sind ($\rightarrow$ keine zusätzliche Information mehr durch die Use-Bits), werden alle Use-Bits zurückgesetzt.

LFU (Least Frequently Used): Bei dieser Strategie wird die Seite mit der geringsten Nutzungshäufigkeit ausgetauscht. Die Nutzungshäufigkeit kann auf verschiedene Art und Weise ermittelt werden. Für diese Aufgabe wird angenommen, dass die Nutzung ab Beginn des Referenzstrings gezählt wird. Ist für mehrere Seiten die gleiche Nutzungshäufigkeit aufgetreten, wird die älteste dieser Seiten zuerst ausgetauscht.

CLIMB (Aufstieg bei Bewährung): Diese Strategie funktioniert prinzipiell wie FIFO, jedoch wird eine Seite, auf die ein erneuter Zugriff erfolgt, während sie bereits im Speicher ist, in der Schlange um eine Position „verjüngt“ (also mit ihrem Nachfolger vertauscht).

OPT (Optimalstrategie): Diese Strategie ersetzt die Seite zuerst, die in Zukunft am längsten nicht mehr benötigt wird. Die Strategie ist in der Praxis nicht realisierbar, weil dazu alle zukünftigen Zugriffe bekannt sein müssen.

Seitenzugriffe werden (im Modell) als *Referenzstring* angegeben, welcher die zeitliche Reihenfolge der angeforderten Seiten enthält. Ein solcher String ist z.B. „01230544351120675231“: Erst erfolgt ein Zugriff auf die Seite 0, dann auf Seite 1, dann auf Seite 2 usw. Seitennummern seien hier immer einstellig!

a) Welche Vor- und Nachteile hat das Paging verglichen mit dem Segmenting?

b) Gegeben sei ein System mit vier Frames. Geben Sie für jede der vorgestellten Strategien die Belegung dieser vier Speicherstellen zu jedem Zeitpunkt sowie die Anzahl der auftretenden Seitenfehler an. Legen Sie dabei den oben angegebenen Referenzstring zugrunde! Zu Beginn sei der Hauptspeicher leer.

c) Unter welcher Voraussetzung kann es sinnvoll sein, bei einem Seitenfehler nicht nur eine sondern gleich mehrere aufeinanderfolgende Seiten (*Prepaging, Look Ahead*) in den Hauptspeicher zu übertragen?

Aufgabe 3: Speicherverwaltung (1+1+2=4 Punkte)

Ein großer Vorteil der virtuellen Speicherverwaltung ist, dass der Programmierer sich nicht um den physikalischen Adressraum kümmern muss, sondern mit logischen Adressen arbeiten kann. Spätestens bei der Ausführung eines Programmes müssen diese logischen Adressen aber in physikalische Adressen umgewandelt werden.

Das Umsetzungsprinzip beim *Segmenting* funktioniert wie folgt: Jedem Programm ist eine Segmenttabelle zugeordnet, in der für jedes logische Segment die physikalische Anfangsadresse `b` und die Größe des Segments `l` eingetragen ist, die das Programmsegment im Speicher einnimmt.

Eine logische Adresse besteht nun aus der Segmentnummer `s`, mit der aus der Segmenttabelle die entsprechende Anfangsadresse `b` bestimmt wird, und einem Offset `d`, der angibt, an welcher Stelle in Relation zur Anfangsadresse sich die Adresse, auf die man zugreifen will, befindet. Dabei wird geprüft, ob der Offset kleiner als die Segmentlänge `l` ist, da anderenfalls eine ungültige Adresse angesprochen wird. Die physikalische Adresse ergibt sich also zu `b+d`.

Für die Speicherverwaltung nach dem Segmentierungsverfahren ist für ein Programm folgende Segmenttabelle gegeben (Länge in Speicherworten):

Segment	Basis	Länge
0	1410	365
1	400	70
2	500	120
3	630	515
4	1145	150

a) Wie viele Speicherworte stehen dem Programm im physikalischen Speicher zur Verfügung?

b) Welches ist die kleinste und welches die größte für das Programm verfügbare physikalische Adresse?

c) Berechnen Sie zu den folgenden physikalischen Adressen jeweils die logischen Adressen:

1. 762
2. 1145
3. 1146
4. 485

Aufgabe 4: Einführung in C (1+1+1+1=4 Punkte)

Diese Aufgabe behandelt die Prioritäten von Operatoren und komplizierte Vereinbarungen in C.

a) In einem Kopierprogramm soll mit folgender **while**-Schleife eine Tastatureingabe eingelesen und kopiert werden:

```
while ( c = getchar () != EOF )
    putchar (c);
```

Warum zeigt dieser Ausdruck nicht das gewünschte Verhalten? Wie sollten Klammern gesetzt werden?

b) Warum ist der folgende Ausdruck problematisch?

```
printf ("%d %d\n", ++n, power(2,n) );
```

c) Beschreiben Sie in Worten was in welcher Reihenfolge in den Zeilen 1 bis 5 passiert.

```
struct {
    int len;
    char *str;
} *p
```

1. `++p -> len;`
2. `p++ -> len;`
3. `*p -> str++;`
4. `(*p -> str)++;`
5. `*p++ -> str;`

d) Worin besteht der Unterschied zwischen `f` und `pf` bzw. zwischen `ap` und `pa`?

1. `int *f();`
2. `int (*pf)()`
3. `int *ap[100];`
4. `int (*pa)[100];`

e) Bonusaufgabe (2 Punkte): Was bedeutet der Ausdruck `float (*(*y[7])())[12];` in Worten? Begründen Sie Ihre Antwort!