Aufgabe 1.1:

a)Aufteilung der Virtuellen Adressraums.

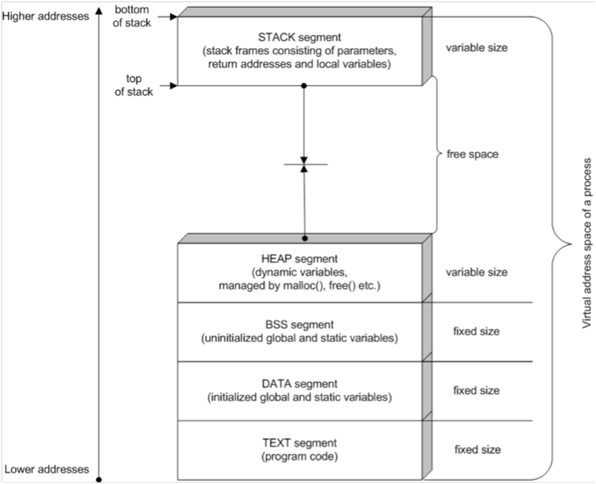


Abbildung 1

Quelle : <http://www.drdobbs.com/security/anatomy-of-a-stack-smashing-attack-and-h/240001832>

b)

Wie auch im oben skizziertes Bild gezeigt wird, werden die **Programmcode** immer in **Textsegment** gespeichert. „Data segment“ beinhaltet alle globale und statische Variablen. Es kann in zwei Segmente unterteilt werden. 1. Initialized 2. Uninitialized (auch bekannt als BSS – Segment).

D.h. dass die **initialisierte Globale Variablen** werden in **Data – Segment** gespeichert.

**Lokale Variablen** intialisiert oder nicht sowie **Funktionsparameter** werden normaler weise auf dem **Stack** gespeichert.[[1]](#footnote-1)

Aufgabe 1.2:

1. Interrupts veranlassen den Prozessor einen gewissen Programmcode auszuführen. Mit einem Interrupt müssen ggf. andere Prozesse pausiert werden, bis die Ereignisbehandlung abgeschlossen ist. Bei der Behandlung wird in den system mode gewechselt. Nach Abschluss des Ereignisbehandlung wird der ggf. anderen Prozess an der Stelle zu Beginn des Interrupts fortgesetzt. [[2]](#footnote-2) Ein Beispiel wäre hierfür das anschließen eines E/A-Geräts dessen Treiber eine Interrupt erzeugt oder bei einem Soundkarte, wenn wieder Sound-Daten zum Abspielen benötigt werden, bevor der Puffer leer wird. [[3]](#footnote-3)
2. Ein Systemaufruf kann sowohl dafür zuständig sein, Informationen an die Hardware, den Kernel selbst oder andere Prozesse zu schicken, als auch solche zu lesen. Dafür ist es manchmal notwendig, Daten aus dem privaten Speicherbereich des Kernels, auf den ein normaler Prozess keinen Zugriff hat, in den Adressbereich eines Prozesses zu kopieren, damit der Prozess auch nach dem Aufruf noch Zugang zu den Daten hat und mit den Ergebnissen seiner Anfrage überhaupt etwas anfangen kann. Auch dies ist eine Folge der strikten Trennung zwischen Kernel-Modus und Benutzer-Modus.[[4]](#footnote-4)

1. Richard Liu, Where are variables stored in c, 2017, Link: <https://www.quora.com/Where-are-variables-stored-in-C> [↑](#footnote-ref-1)
2. Mandl, P. (2014). *Grundkurs Betriebssysteme [Elektronische Ressource] : Architekturen, Betriebsmittelverwaltung, Synchronisation, Prozesskommunikation, Virtualisierung* (4. Aufl.). Wiesbaden: Springer Fachmedien Wiesbaden. S.51 [↑](#footnote-ref-2)
3. Mandl, P. (2014). *Grundkurs Betriebssysteme [Elektronische Ressource] : Architekturen, Betriebsmittelverwaltung, Synchronisation, Prozesskommunikation, Virtualisierung* (4. Aufl.). Wiesbaden: Springer Fachmedien Wiesbaden. S. 269 [↑](#footnote-ref-3)
4. <https://de.wikipedia.org/wiki/Systemaufruf#cite_note-3> [↑](#footnote-ref-4)