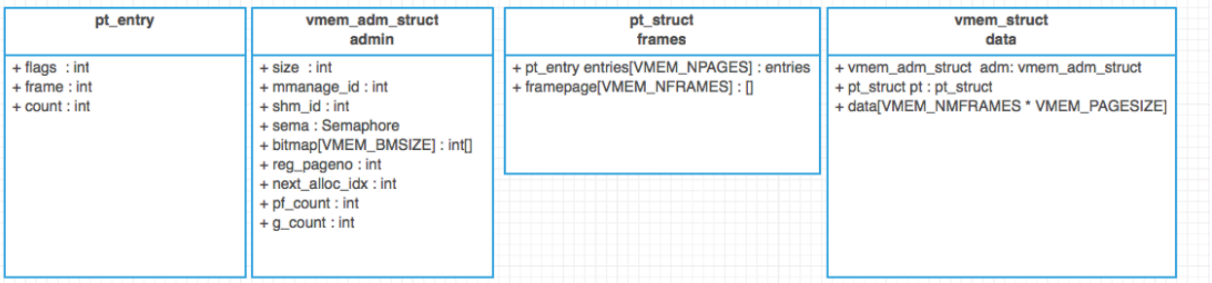
1. **Visualisierung der geschachtelten Strukturen und Parameter Quantifizierung: vmem_struct; pt_struct; pt_entry; vm_adm_struct**



2. **Kennzeichnung in welcher Reihenfolge die jeweiligen Funktionen zugreifen. (Wir haben es so verstanden, dass es sich auf die Struktur bezieht).**

Siehe Sequenzdiagramm.

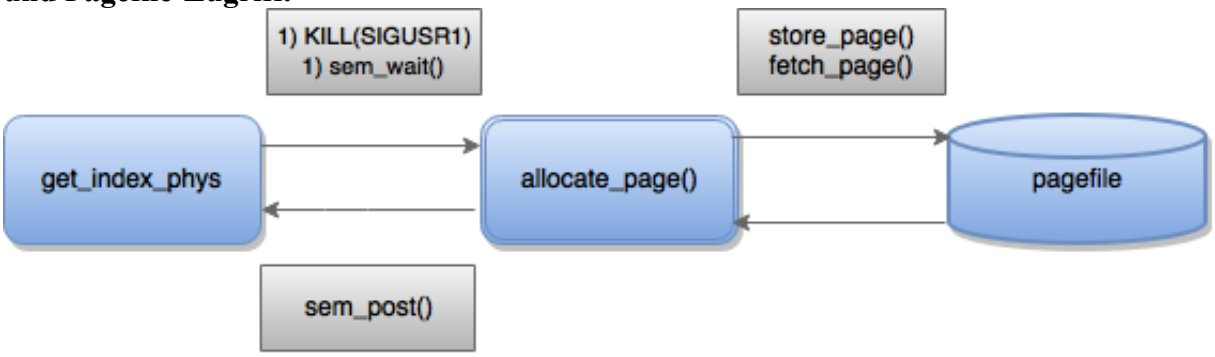
3. **Welcher Speicherbereich fehlt im Übersichtsbild?**

Unser Hauptspeicher wird visualisiert durch pagefile.bin.

4. **Vereinfachen Sie die Bitmap.**

Wir können die Bitmap so vereinfachen, indem wir eine Variable anstatt der Bitmap verwenden. Diese Variable wird dementsprechend bei der Allokation von den ersten 16 Seiten verwendet, um bei jeder einzelnen Allokation wird die Page anstatt in einen Frame in die Variable geladen. Bei jeder Allokation wird die Variable inkrementiert und sollte die Variable beziehungsweise unser Zähler den Wert 15 erreichen, soll der CLOCK-Algo für weitere Allokationen verwendet werden.

5. **Skizzieren Sei ein Kommunikationsrahmen mit Signalisierung, Semaphoren-Zugriff und Pagefile-Zugriff.**



Vmappl wird gestartet, mmanage wartet auf ein asynchrones Signal, danach versucht vmappl die Page zu lesen. Ist die Page nicht in einem Frame wird der erste Pagefault erstellt. Wenn die Page vorhanden ist wird sie von vmappl gelesen oder beschrieben. Danach wartet vmappl durch sem_wait auf den sem_post von mmanage. Währenddessen sucht mmanage nach einem leeren Frame. Gibt es einen leeren wird dieser Frame beschrieben, falls es keinen leeren gibt sucht CLOCK nach einer Page welche zu ersetzen ist. Wurde die Page gefunden kann sie mit der angeforderten überschrieben werden. Danach werden die Bit-Flags überprüft und aktualisiert. Danach kriegt vmappl ein Signal von mmanage durch sem_post um aufzuwachen.

6. **Wie wird die geforderte Seite bestimmt? Fkt. = ?**

Durch get_phys_index(..) wird die geforderte Seite bestimmt.

7. **Erklären Sie die Parameter der Pagefile-Zugriffe mit fwrite() u. fread(). Was liefert/erzeugt fseek() u. wie wird der variable Eingangsparameter offset betstimmt?**

- **fwrite(Anfangsadresse, Größe in Bytes, Anzahl der Elemente, Pfadname der Zieldatei)**: schreibt ein Datenarray mit einer gegebenen Anzahl von Elementen von einer definierten Größe aus dem Speicher an die aktuelle Position im angegebenen Stream.
- **fread(Anfangadresse, Größe in Bytes, Anzahl der Elemente, Pfadname der Zieldatei)**: analog zu fwrite, jedoch nur lesen.
- fseek(): setzt die aktuelle schreibe/lese-Position in der angegebenen Datei zu einem definierten Bezugspunkt.
- offset: berechnet sich aus der Größe der req_pageno * der Größe des INT wertes (4Bytes) * Pagesize(default = 8).

8. **Geben Sie die Aufrufreihenfolge der Fkt. aus manage.h/allocate_page.c an und nennen deren Eingangs- /Ausgangsparameter.**

Zuerst wird die Funktion find_remove_frame(void) aufgerufen. Danach wird, wenn die Seite modifiziert wurde, store_page(removed_page_id) (also mit zu löschenden Seite als Eingangsparamentern) mit keinem Ausgangsparameter. update_pt(free_sprace_in_bitmap) aktualisiert die freie Stelle mit dem Eingangsparameter, der von find_remove_frame (ansonsten -1) kommt. Danach ruft fetch_page(reg_page_no) mit dem Eingangsparameter von der angeforderten Seite. Dach wird die Semaphore freigegeben.

10. **Welche Fkt. setzen und lesen die Status-Bits: Present, Modified/Dirty, Used (Tabelle zur Übersicht)?**

Die Funktion update_pt setzt die Present, Dirty und Used Bits.

11. **Was zählt der Zähler adm.g_count, welche Fkt. inkrementiert ihn und wie/wo wird er genutzt?**

Der Zähler beschreibt den Zugriff auf den Speicher (wichtig für den LRU-Algorithmus). adm.g_count wird in der Funktion get_phys_index inkrementiert. Wird in der Funktion logger benutzt und in weiteren Funktionen wie Update Pagetable, Allocate Page, Dump_pt(Ausgabe des Pagetables).

12. **Von welcher Fkt. wird der Zähler pf_count inkrementiert?**

In der Funktion allocate_page().