Structures de données élémentaires dans le contexte du TP #1

Liste chaînée (Linked List)

- La liste chaînée permet d'insérer un élément dans une liste ordonnée d'éléments. L'ajout peut se faire n'importe où.

- On peut retirer n'importe lequel des éléments de la liste à tout moment.

Liste chaînée (suite) et utilisation dans le TP #1
Exemple de fichier de transactions (trans2.txt)

```
1  A
2  L 1-4
3  A
4  L 1-4
5  A
6  L 1-4
7  E 2
8  L 1-4
9  A
10 L 1-4
11 X 1 Mult-Numbers.olc3
12 X 2 Mult-Numbers.olc3
13
```

Liste chaînée (suite)
(Voir fichier gestionListeChaineeVMS.h)

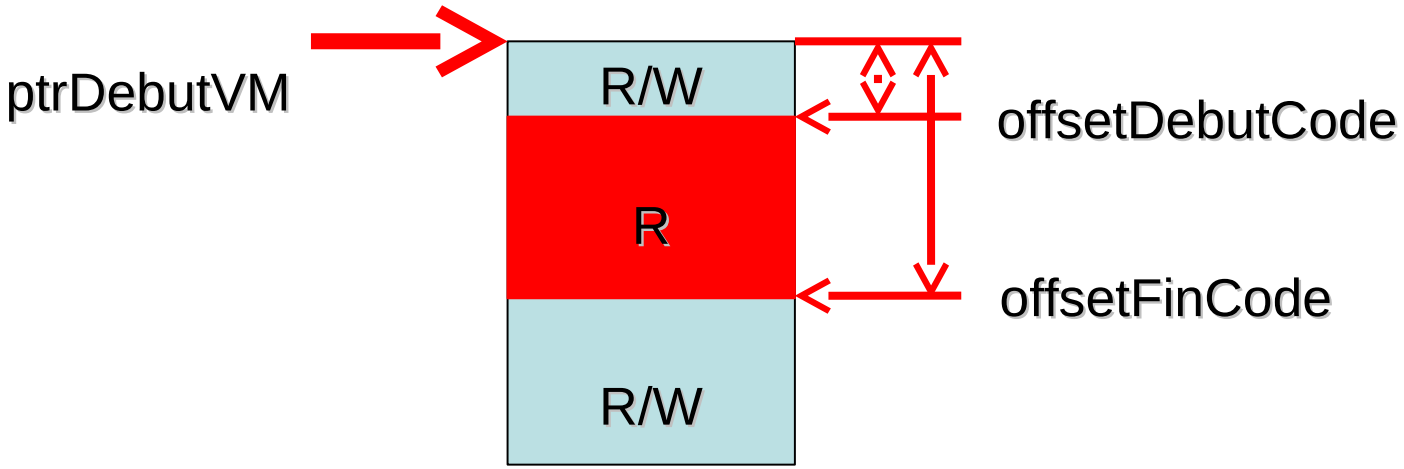
- Structure noeudVM

```
14  struct noeudVM{
15      struct infoVM  VM;
16      struct noeudVM    *suivant;
17      };
18
```

Liste chaînée (suite)
(Voir fichier gestionListeChaineeVMS.h)

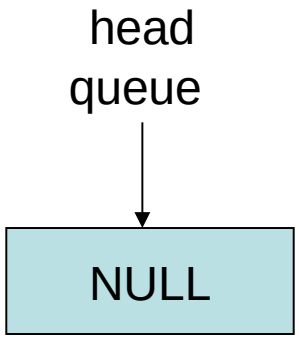
- Structure infoVM

```
18 struct infoVM{
19     int       noVM;
20     unsigned char  busy;
21     uint16_t *  ptrDebutVM;
22     uint16_t    offsetDebutCode; // region memoire ReadOnly
23     uint16_t    offsetFinCode;
24     };
25
```



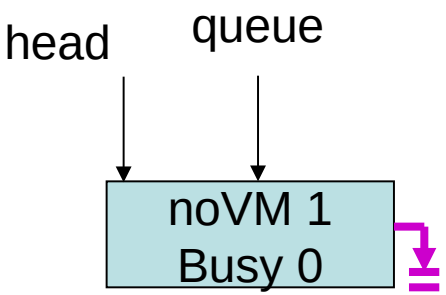
Liste chaînée (suite)
(Voir fichier gestionVMS_MAIN.c)

- Initialisation des pointeurs de début et fin de la liste chaînée de machines virtuelles (VM) (voir ***main()***)



Liste chaînée (suite) (Voir fichier gestionListeChaineeVMS.c)

- Ajout d'une VM dans la liste chaînée de VM (première VM) (voir ***addItem()***)

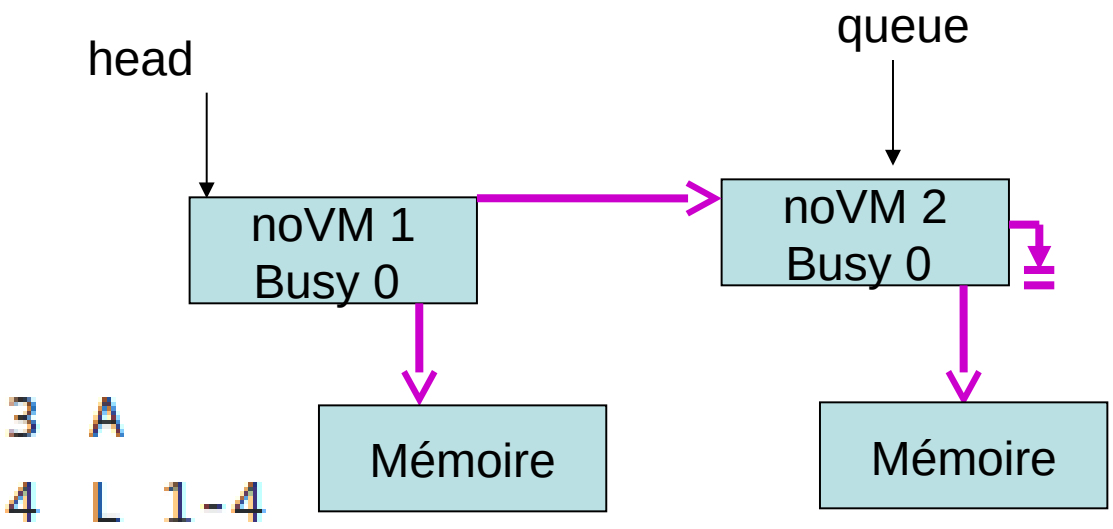



```
1 A
2 L 1-4
```

```
noVM  Busy?            Adresse Debut VM
=========================================================
1        0       0x7fecf941c010
=========================================================
```

Liste chaînée (suite)
(Voir fichier gestionListeChaineeVMS.c)

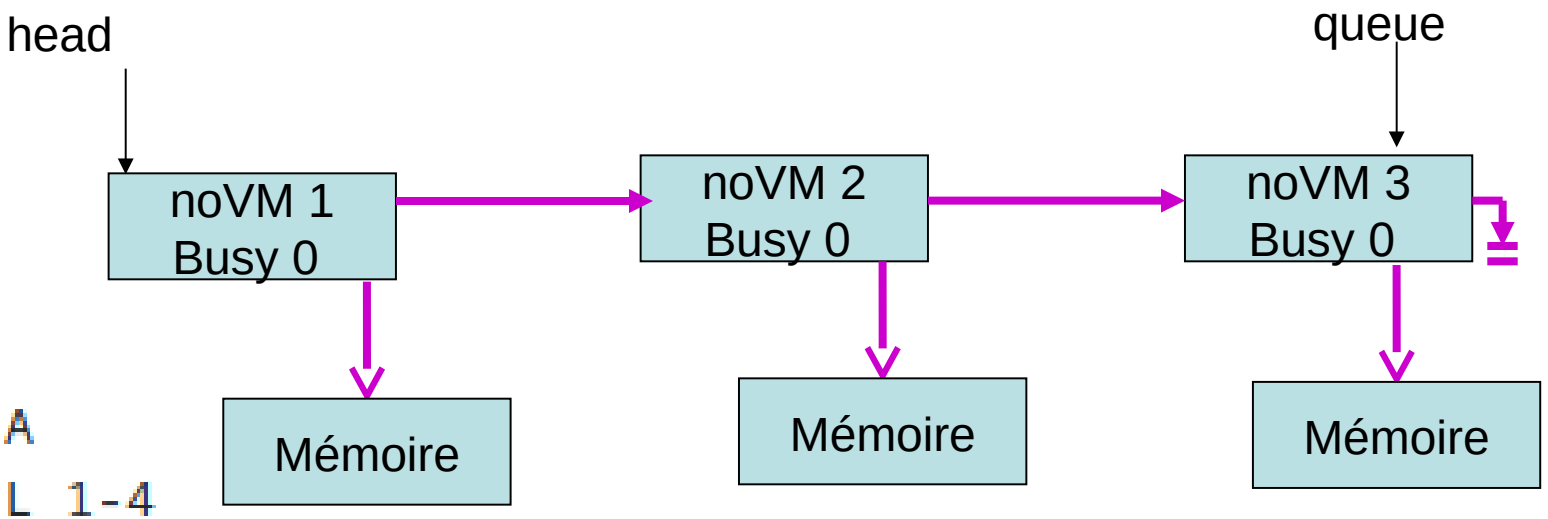
- Ajout d'une VM dans la liste chaînée de VM (seconde VM) (voir ***addItem()***)




```
noVM  Busy?              Adresse Debut VM
=============================================================
1         0         0x7f74e5ace010
2         0         0x7f74e5aad010
=============================================================
```

Liste chaînée (suite)
(Voir fichier gestionListeChaineeVMS.c)

- Ajout d'une VM dans la liste chaînée de VM (troisième VM) (voir ***addItem()***)

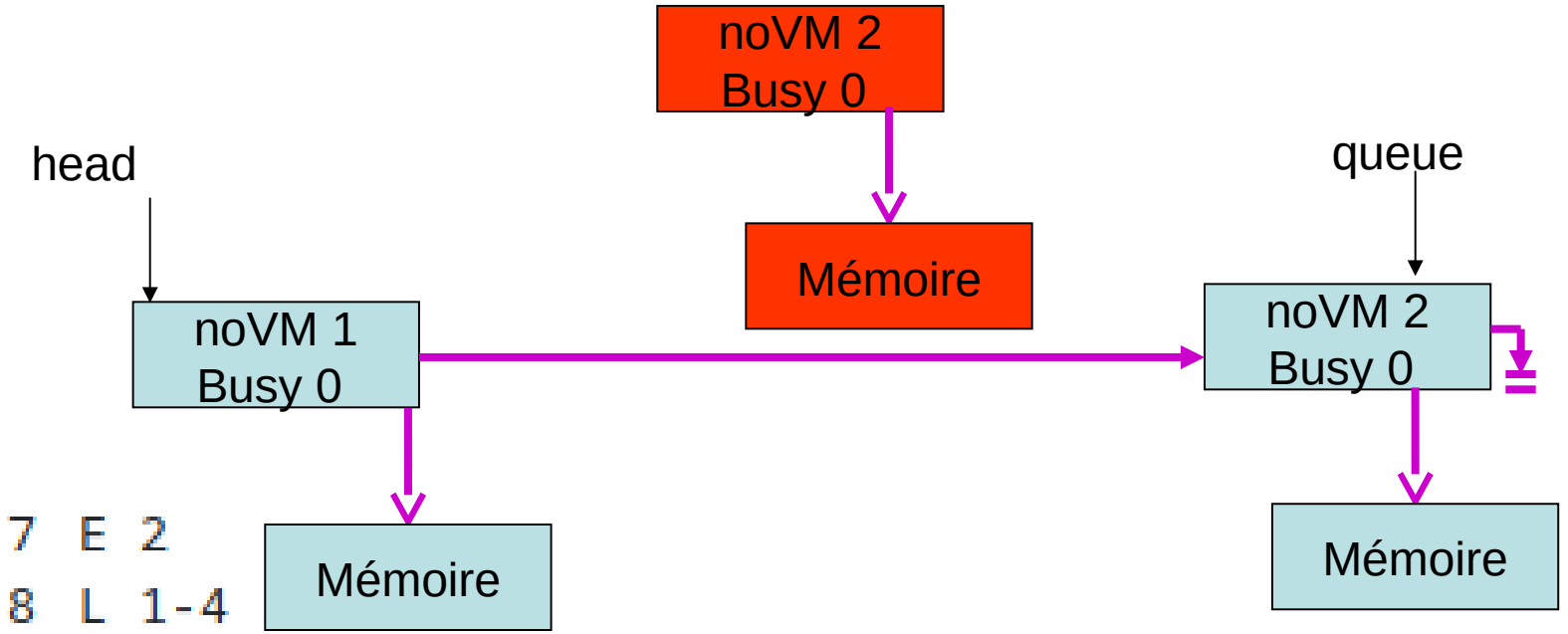



```
5  A
6  L  1-4
```

```
noVM   Busy?              Adresse Debut VM
=============================================================
1          0         0x7f74e5ace010
2          0         0x7f74e5aad010
3          0         0x7f74e5a8c010
=============================================================
```

Liste chaînée (suite) (Voir fichier gestionListeChaineeVMS.c)

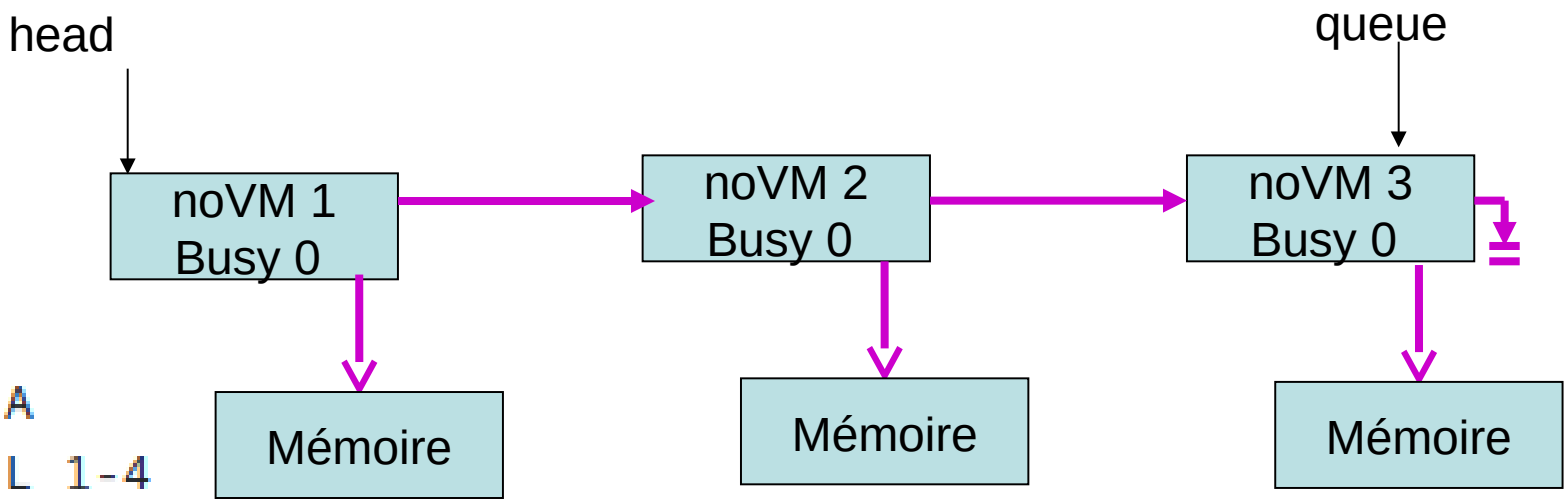
- Effacement du noeud 2 dans la liste chaînée de VM




```
noVM   Busy?                Adresse Debut VM
=============================================================
1          0          0x7f74e5ace010
2          0          0x7f74e5a8c010
=============================================================
```

Liste chaînée (suite)
(Voir fichier gestionListeChaineeVMS.c)

- Ajout d'une version dans la liste chaînée de VM (troisième VM) (voir ***addItem()***)




```
9  A
10 L  1-4
```

```
noVM   Busy?              Adresse Debut VM
=============================================================
1          0          0x7f74e5ace010
2          0          0x7f74e5a8c010
3          0          0x55a3ed639900
=============================================================
```

Liste chaînée en code (findItem()) (Voir fichier gestionListeChaineeVMS.c)

```
25  //# Recherche un noeud dans la liste chaînée
26  //# ENTREE: Numéro de la VM
27  //# RETOUR: Un pointeur vers le noeud (VM)) recherché
28  //#      NULL si le noeud est introuvable
29  struct noeudVM * findItem(const int no){
30      //La liste est vide
31      if ((head==NULL)&&(queue==NULL)) return NULL;
32      //Pointeur de navigation
33      struct noeudVM * ptr = head;
34      if(ptr->VM.noVM==no) // premier noeudVM
35          return ptr;
36      //Tant qu'un noeud suivant existe
37      while (ptr->suivant!=NULL){
38          //Déplacement du pointeur de navigation
39          ptr=ptr->suivant;
40
41          //Est-ce l'item recherché?
42          if (ptr->VM.noVM==no){
43              return ptr;
44          }
45      }
46      //On retourne un pointeur NULL
47      return NULL;
48  }
```

Liste chaînée en code (findPrev()) (Voir fichier gestionListeChaineeVMS.c)

```
50  //#########################################
51  //# Recherche le PRÉDÉCESSEUR d'un noeud dans la liste chaînée
52  //# ENTREE: Numéro du noeud (VM)) a supprimer
53  //# RETOUR: Le pointeur vers le prédécesseur est retourné
54  //#      NULL si le noeud est introuvable
55  struct noeudVM * findPrev(const int no){
56      //La liste est vide
57      if ((head==NULL)&&(queue==NULL)) return NULL;
58      //Pointeur de navigation
59      struct noeudVM * ptr = head;
60      //Tant qu'un noeud suivant existe
61      while (ptr->suivant!=NULL){
62          //Est-ce le prédécesseur du noeud recherché?
63          if (ptr->suivant->VM.noVM==no){
64              //On retourne un pointeur sur l'item précédent
65              return ptr;
66          }
67          //Déplacement du pointeur de navigation
68          ptr=ptr->suivant;
69      }
70      //On retourne un pointeur NULL
71      return NULL;
72  }
```

Liste chaînée en code (addItem()) (Voir fichier gestionListeChaineeVMS.c)

```
74  //#########################################################
75  //# Ajouter un noeud (VM) a la fin de la liste chaînée de VM
76  //# ENTREE:
77  //#   RETOUR:
78  void addItem(){
79      //Création de l'enregistrement en mémoire
80      struct noeudVM* ni = (struct noeudVM*)malloc(sizeof(struct noeudVM));
81      //Affectation des valeurs des champs
82      ni->VM.noVM = ++nbVM;
83      ni->VM.busy = 0;
84      ni->VM.ptrDebutVM = (unsigned short*)malloc(sizeof(unsigned short)*65536);
85
86      if ((head == NULL) && (queue == NULL)){//liste VM vide
87        ni->suivant= NULL;
88        queue = head = ni;
89        return;
90      }
91      struct noeudVM* tptr = queue;
92      ni->suivant= NULL;
93      queue = ni;
94      tptr->suivant = ni;
95  }
```

Liste chaînée en code (removeItem()) (Voir fichier gestionListeChaineeVMS.c)

```
97   //#########################################
98   //# Retire un item de la liste chaînée
99   //# ENTREE: noVM: numéro du noeud (VM) a retirer
100  void removeItem(const int noVM){
101     struct noeudVM * ptr;
102     struct noeudVM * tptr;
103     struct noeudVM * optr;
104     //Vérification sommaire (noVM>0 et liste non vide)
105     if ((noVM<1)||((head==NULL)&&(queue==NULL)))
106        return;
107     //Pointeur de recherche
108     if(noVM==1){
109        ptr = head; // suppression du premier element de la liste de VM
110     }
111     else{
112        ptr = findPrev(noVM); // ptr sur le noeud (VM) precedent
113     }
```

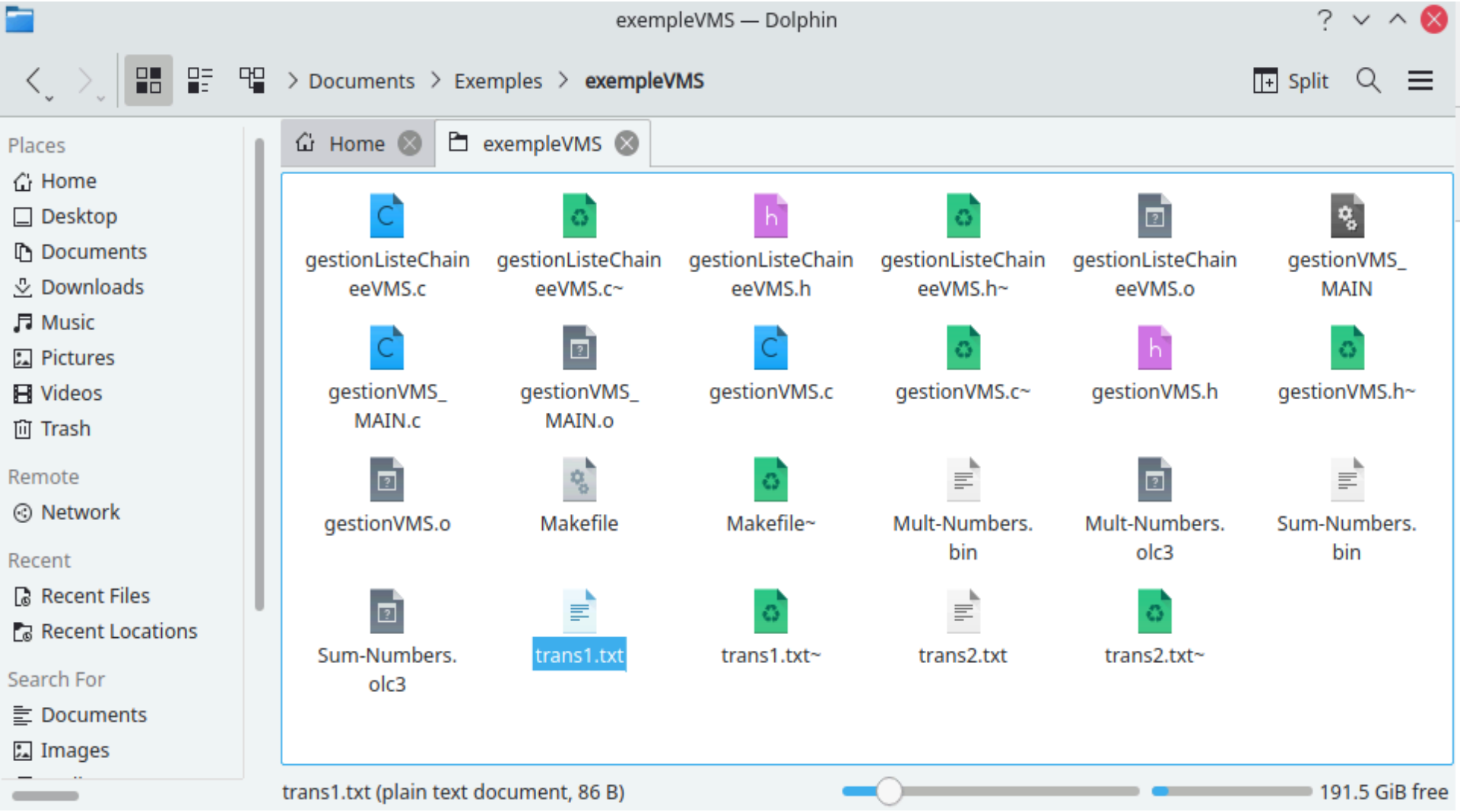
Liste chaînée en code (removeItem()) (Voir fichier gestionListeChaineeVMS.c) (suite …)

```
114     // Noeud (VM) trouvé
115     if (ptr!=NULL){
116        nbVM--;
117        // Memorisation du pointeur de l'item en cours de suppression
118        // Ajustement des pointeurs
119        if((head == ptr) && (noVM==1)) // suppression de l'element de tete
120        {
121           if(head==queue) // un seul element dans la liste
122           {
123              free(ptr->VM.ptrDebutVM);
124              free(ptr);
125              queue = head = NULL;
126              return;
127           }
128           tptr = ptr->suivant;
129           head = tptr;
130           free(ptr->VM.ptrDebutVM);
131           free(ptr);
132        }
```

Liste chaînée en code (removeItem()) (Voir fichier gestionListeChaineeVMS.c) (suite …)

```
133         else if (queue==ptr->suivant) // suppression de l'element de queue
134         {
135             queue=ptr;
136             free(ptr->suivant->VM.ptrDebutVM);
137             free(ptr->suivant);
138             ptr->suivant=NULL;
139             return;
140         }
141         else // suppression d'un element dans la liste
142         {
143             optr = ptr->suivant;
144             ptr->suivant = ptr->suivant->suivant;
145             tptr = ptr->suivant;
146             free(optr->VM.ptrDebutVM);
147             free(optr);
148         }
149         while (tptr!=NULL){ // ajustement des numeros de VM
150             tptr->VM.noVM--;
151         //Déplacement du pointeur de navigation
152             tptr=tptr->suivant;
153         }
154     }
155 }
```

Application gestionVMS séquentielle



Fichier gestionVMS_MAIN.c

```
12 //#####################################
13
14 #include "gestionListeChaineeVMS.h"
15 #include "gestionVMS.h"
16
17 //Pointeur de tête de liste
18 struct noeud* head;
19 //Pointeur de queue de liste pour ajout rapide
20 struct noeud* queue;
21 // nombre de VM actives
22 int nbVM;
23
24
25 int main(int argc, char* argv[]){
26
27     //Initialisation des pointeurs
28     head = NULL;
29     queue = NULL;
30     nbVM = 0;
31
32     readTrans(argv[1]); // lecture du fichier de transaction
33
34     //Fin du programme
35     exit( 0);
36 }
```

Fichier gestionVMS.c (Fonction readTrans())

```
476 //#########################################
477 //#
478 //# fonction utilisée pour le traitement  des transactions
479 //# ENTREE: Nom de fichier de transactions
480 //# SORTIE:
481 void* readTrans(char* nomFichier){
482     FILE *f;
483     char buffer[100];
484     char *tok, *sp;
485 
486     //Ouverture du fichier en mode "r" (equiv. "rt") : [r]ead [t]ext
487     f = fopen(nomFichier, "rt");
488     if (f==NULL)
489         error(2, "readTrans: Erreur lors de l'ouverture du fichier.");
490 
491     //Lecture (tentative) d'une ligne de texte
492     fgets(buffer, 100, f);
493 
```

Fichier gestionVMS.c (Fonction readTrans()) (suite …)

```
495    while(!feof(f)){
496
497        //Extraction du type de transaction
498        tok = strtok_r(buffer, " ", &sp);
499
500        //Branchement selon le type de transaction
501        switch(tok[0]){
502            case 'A':
503            case 'a':{
504                //Appel de la fonction associée
505                addItem(); // Ajout de une VM
506                break;
507                }
508            case 'E':
509            case 'e':{
510                //Extraction du paramètre
511                int noVM = atoi(strtok_r(NULL, " ", &sp));
512                //Appel de la fonction associée
513                removeItem(noVM); // Eliminer une VM
514                break;
515                }
```

Fichier gestionVMS.c (Fonction readTrans()) (suite …)

```
495    while(!feof(f)){
496
497        //Extraction du type de transaction
498        tok = strtok_r(buffer, " ", &sp);
499
500        //Branchement selon le type de transaction
501        switch(tok[0]){
502            case 'A':
503            case 'a':{
504                //Appel de la fonction associée
505                addItem(); // Ajout de une VM
506                break;
507                }
508            case 'E':
509            case 'e':{
510                //Extraction du paramètre
511                int noVM = atoi(strtok_r(NULL, " ", &sp));
512                //Appel de la fonction associée
513                removeItem(noVM); // Eliminer une VM
514                break;
515                }
```

Fichier gestionVMS.c (Fonction readTrans()) (suite …)

```
516         case 'L':
517         case 'l':{
518            //Extraction des paramètres
519            int nstart = atoi(strtok_r(NULL, "-", &sp));
520            int nend = atoi(strtok_r(NULL, " ", &sp));
521            //Appel de la fonction associée
522            listItems(nstart, nend); // Lister les VM
523            break;
524            }
525         case 'X':
526         case 'x':{
527            //Appel de la fonction associée
528            int noVM = atoi(strtok_r(NULL, " ", &sp));
529            char *nomfich = strtok_r(NULL, "\n", &sp);
530            executeFile(noVM, nomfich); // Executer le code binaire du fichier nomFich sur la VM noVM
531            break;
532            }
533      }
534      //Lecture (tentative) de la prochaine ligne de texte
535      fgets(buffer, 100, f);
536   }
537   //Fermeture du fichier
538   fclose(f);
539   //Retour
540   return NULL;  }
```

Création du fichier exécutable gestionVMS_MAIN (Makefile)

```
1  gestionVMS_MAIN: gestionVMS_MAIN.o gestionListeChaineeVMS.o gestionVMS.o
2      gcc -o gestionVMS_MAIN gestionVMS_MAIN.o gestionListeChaineeVMS.o gestionVMS.o
3  gestionVMS_MAIN.o: gestionVMS_MAIN.c gestionListeChaineeVMS.h gestionVMS.h
4      gcc -c gestionVMS_MAIN.c -Wall -I.
5  gestionListeChaineeVMS.o: gestionListeChaineeVMS.c gestionListeChaineeVMS.h gestionVMS.h
6      gcc -c gestionListeChaineeVMS.c -Wall -I.
7  gestionVMS.o: gestionVMS.c gestionListeChaineeVMS.h gestionVMS.h
8      gcc -c gestionVMS.c -Wall -I.
9  clean:
10     rm *.o
```

Fichier de transactions (trans2.txt)

```
1  A
2  L 1-4
3  A
4  L 1-4
5  A
6  L 1-4
7  E 2
8  L 1-4
9  A
10 L 1-4
11 X 1 Mult-Numbers.olc3
12 X 2 Mult-Numbers.olc3
```

Exécution du fichier gestionVMS_MAIN

```
noVM  Busy?              Adresse Debut VM
=========================================================
1         0         0x7f9e0d61e010
=========================================================

noVM  Busy?              Adresse Debut VM
=========================================================
1         0         0x7f9e0d61e010
2         0         0x7f9e0d5fd010
=========================================================

noVM  Busy?              Adresse Debut VM
=========================================================
1         0         0x7f9e0d61e010
2         0         0x7f9e0d5fd010
3         0         0x7f9e0d5dc010
=========================================================
```

```
1 A
2 L 1-4

3 A
4 L 1-4

5 A
6 L 1-4
```

Exécution du fichier gestionVMS_MAIN

```
noVM  Busy?              Adresse Debut VM
=========================================================
1         0         0x7f9e0d61e010
2         0         0x7f9e0d5dc010
=========================================================

noVM  Busy?              Adresse Debut VM
=========================================================
1         0         0x7f9e0d61e010
2         0         0x7f9e0d5dc010
3         0         0x55a4a41f6900
=========================================================


 add reg[r0] (sum) = 10
 add reg[r0] (sum) = 20
 add reg[r0] (sum) = 30
 HALT

 add reg[r0] (sum) = 10
 add reg[r0] (sum) = 20
 add reg[r0] (sum) = 30
 HALT
```

```
7  E 2
8  L 1-4
```

```
9  A
10 L 1-4
```

```
11 X 1 Mult-Numbers.olc3
12 X 2 Mult-Numbers.olc3
```