1. Que fait le programme suivant ?
2. Décrivez le code et son fonctionement en détail.
3. A l'aide d'un schema expliquez ce que fait l'appel à la fonction mmap.
4. Que deviens la mémoire partagée lors de l'appel aux fonctions fork ? Expliquez par un schema.
5. Ce programme peut ne pas fonctionner dans le cas de l'appel suivant:

   sharedmem monfichier a b b d

   pourquoi ?
6. Quel serait l'effet d'utiliser MAP_PRIVATE au lieu de MAP_SHARED dans la fonction mmap ? Cela permet-il de resoudre le problème ci-dessus ?
7. Est-ce toujours un avantage d'avoir plusieurs processus qui travaillent parallèlement sur une mémoire partagée plutôt que un processus travaillant séquentiellement ? Pourquoi ?

sharedmem.c

```c
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

void OnError(char *str)
{
    perror(str);
    exit(EXIT_FAILURE);
}

int processFile(char* fileContent, off_t fileSize, char replaceChar,
    char byChar)
{
    int i;

    printf("Processing char %c to be replaced by %c\n", replaceChar,
        byChar);
    for(i=0; i < fileSize; i++)
        if(fileContent[i] == replaceChar)
            fileContent[i] = byChar;

    return 0;
}

int main(int argc, char* argv[])
{
```

```
30
31      int fd, iArg;
32      pid_t fork_res;
33      struct stat fileStat;
34      char* fileContent;
35
36      if((argc % 2) || (argc < 4))
37      {
38          printf("Wrong number of arguments: %d\n\n",argc);
39          printf("Usage:\n\t sharedmem filename replaceChar byChar [
                replaceChar2 byChar2 [replaceChar3 byChar3 [...]]]\n\n");
40          exit(EXIT_SUCCESS);
41      }
42
43      if((fd = open(argv[1], O_RDWR)) == -1)
44          OnError("Cannot open file");
45
46      if( fstat(fd, &fileStat) == -1)
47          OnError("fstat");
48
49      fileContent = (char*) mmap(NULL, (size_t) fileStat.st_size,
            PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
50      if(fileContent == MAP_FAILED)
51          OnError("mmap");
52      close(fd);
53
54      for(iArg=2;iArg<argc;iArg+=2)
55      {
56          fork_res = fork();
57          if(fork_res == -1)
58              OnError("Fork");
59          else if(fork_res == 0)
60              return processFile(fileContent ,fileStat.st_size, *argv[iArg
                    ], *argv[iArg+1]);
61      }
62
63      for(iArg=2;iArg<argc;iArg+=2)
64          wait(NULL);
65
66      if(munmap(fileContent, fileStat.st_size) == -1)
67          OnError("munmap");
68
69      return 0;
70  }
```

makefile

```
1  all: sharedmem
2
3  sharedmem: sharedmem.c
4          gcc sharedmem.c -Wall -o sharedmem
```