Задачи за подготовка върху системни примитиви за работа с

обикновени файлове

1. Да се напише програма, която има функционалността на **wc c**.

Програмата може да приема 0 или повече на брой аргументи от

командния ред, като ако броят е 0, файлът, за който броим, е

стандартния вход.

2. Да се напише програма, която има функционалността на **wc l**.

Програмата може да приема 0 или повече на брой аргументи от

командния ред, като ако броят е 0, файлът, за който броим, е

стандартния вход.

3. Да се напише програма, която има функционалността на **wc** с

възможни опции **c,l,w** и да приема като аргументи 0 или повече

имена на файлове.

4. Да се напише програма, която има функционалността на

**tee [FILE]**

5. Да се напише програма, която има функционалността на

**cat n [FILE]**

6. Да се напише програма, която има функционалността на

**paste FILE1 FILE2**

7. Да се напише програма, която има функционалността на

**cut --bytes=N-M FILE**

Задачи за упражнение на системни примитиви за работа с процеси

1. Да се напише програма, която отваря един файл (*bigbag*) в режим

само за четене и 4 празни файла (*smallbags[0], …, smallbags[3]*) в

режим само за писане. Имената на тези файлове са аргументите от

командния ред.

Нашата програма създава 4 процеса-деца. i-тото дете има отворен

само *smallbags[i]* в режим само за четене.

На всяка секунда нашата програма (родителят) си избира по едно

произволно живо дете (номер *I*) и проверява дали то не е изчело

всичко, записано до момента в *smallbags[I]*. Ако това е така – изчаква

завършването на детето. Ако това не е така, прочита *N* байта от

*bigbag* и ги записва в *smallbags[I]*.

На всеки пет секунди всяко дете опитва да прочете по *N* или *2N* байта

от своя файл, като ако файлът е празен, детето завършва.

Процесът родител трябва да изпише в какъв ред са погребани децата

му.

*За произволния избор:* ***man 3 rand***

2. Да се напише програма, която приема един аргумент от командния

ред, който представлява валидно име на файл. Файлът съдържа

няколко реда, на всеки от които има по 2 думи – първата е собствено

име на изпълним файл, втората е име на обикновен файл (който

може и да не съществува). Програмата пуска процес, който трябва да

бъде дете на init и периодично (през 2 секунди) да прочита по 1 ред от

указания файл и да пуска за изпълнение програмата, указана от

първата дума на ред, като пренасочва стандартния й изход към

файлът, указан от втората дума на реда.

Примерни тестови задачи:

1. Какво ни връща при успех POSIX системния примитив **dup2**?

2. Какво се подава като втори аргумент на POSIX системния примитив

**open**?

3. Как трябва да извикаме POSIX системния примитив **creat**, за да

създадем файл с име *garga* в текущата директория, който да има

права за достъп *rw-r---w-*?

4. С кой POSIX системен примитив можем да изчакаме завършването на

процес-дете и да освободим от Таблицата на процесите записа за

него?

5. Кои от следните са имена на POSIX системни примитиви, кои са

имена на C библиотечни функции и кои – нито едно от двете?

**fork sprintf open memmove pop sleep**

6. Кое от следните може да бъде върната стойност от **execlp**?

А) 0 Б) 1 В) -1 Г) 255

7. Текущата позиция за отварянето на файл през файлов дескриптор *fd*

е 0. След кое от следните извиквания на **lseek** тя ще е различна?

(Приемаме, че *fd* е асоцииран с

seekable файл и текущата позиция

няма да се променя от някой друг процес)

А) **lseek(fd,1,SEEK\_SET);**

Б) **lseek(fd,0,SEEK\_CUR);**

В) **lseek(fd,lseek(fd,0,SEEK\_END),SEEK\_END);**

Г) **lseek(fd,lseek(fd,0,SEEK\_END),SEEK\_END);**

Задачи с fork, exec, wait, exit

Задача 1 (croncho). Да се напише програма, която приема като първи

аргумент число seconds, като втори аргумент име на изпълним файл и като

следващи аргументи аргументи за изпълнимия файл. Нашата програма

трябва през seconds секунди да пуска (в свой процес-дете) указаната

програма с указаните аргументи.

Задача 2 (smart\_croncho). Да се напише програма, която приема като първи

име на изпълним файл и като следващи аргументи аргументи за

изпълнимия файл. Нашата програма трябва да пусне (в свой процес-дете)

указаната програма с указаните аргументи и да следи за това, когато

детето приключи, да го пусне отново.

Задача 3 (splitter). Да се напише програма, която прочита един ред от

стандартния вход, разделя го на думи и execute-ва програмата, указана от

първата дума с аргументи, указани от останалите думи.

Максимална дължина на ред: 1024 символа.

*Жокер:* ***man 3 strtok***

Задача 4 (shelly). Да се напише програма, която прочита един ред от

стандартния вход, разделя го на думи и execute-ва програмата, указана от

първата дума с аргументи, указани от останалите думи, в свой процес-

дете. След приключване на детето, отново прочита един ред и т.н.

По желание може да се добави функционалност, ако от стандартния вход

сме прочели exit, програмата ни да завършва.