МОДЕЛИРОВАНИЕ РАБОТЫ ИНТЕРПРЕТАТОРА SHELL
(программа My_Shell)

Интерпретатор команд SHELL — это программа, воспринимающая и выполняющая команды, вводимые пользователем с терминала или находящиеся в командном файле (т.е. текстовом файле, содержащем последовательность команд). Команды делятся на «внутренние», которые SHELL выполняет непосредственно, и «внешние», для выполнения
которых создаются отдельные процессы. Имя любого исполняемого файла UNIX является «внешней» командой SHELL. Кроме того, SHELL позволяет соединять выполняющиеся команды каналами, перенаправлять ввод-вывод команд в файлы, выполнять команды в асинхронном режиме.

Синтаксис команд SHELL (за исключением управляющих команд типа "if", "for" и некоторых других) определяется следующими правилами:

⟨команда_SHELL⟩::= ⟨список_команд⟩

⟨список_команд⟩::= ⟨конвейер⟩ { [один из **&**, **&&**, **||**, **;**] ⟨конвейер⟩} [**&**, **;**]

⟨конвейер⟩::= ⟨команда⟩ { **|** ⟨команда⟩}

⟨команда⟩::= ⟨простая_команда⟩ |
(⟨список_команд⟩**)** [**<**⟨имя_файла⟩][[один из **>**, **>>**] ⟨имя_файла⟩]

⟨простая_команда>::=⟨имя_файла⟩ {⟨аргумент⟩}
[**<**⟨имя_файла⟩] [[один из **>**, **>>**] ⟨имя_файла⟩]

Аргумент команды — это имя файла, флаг, ключ и т. пр.
Конструкция "[один из ...]" означает обязательное присутствие одного (и только одного) из перечисленных в ней элементов.

Операции >, >> позволяют перенаправить в файл результаты работы любой команды, предназначенные для стандартного вывода. (Операция >> добавляет результаты в конец указанного файла). Если файл не существовал, то он создается. Операция < переназначает входной поток, так что данные, которые выполняемая команда должна считывать из входного потока, будут взяты из файла.

Через точку с запятой (;) указываются последовательно выполняемые команды (конвейеры). Символ && (||) означает, что следующая за ним команда будет выполнена только в том случае, если предыдущая команда завершилась успешно (неуспешно), т.е. возвратила нулевое (ненулевое) значение.

Операция & указывает, что предшествующая ей команда (конвейер) выполняется в фоновом режиме, т.е. SHELL не ожидая завершения данной команды выполняет следующую.

Конвейер позволяет запустить указанные в нем команды в виде отдельных процессов, соединив каналом стандартный выходной поток каждого процесса (кроме последнего) со стандартным входным потоком следующего процесса. Работа конвейера завершается с окончанием работы последней команды в нем.

Круглые скобки означают, что для выполнения указанного в них списка команд создается новый процесс SHELL, который исполнит эти команды и возвратит в отцовский процесс статус (код) завершения последней выполненной команды.

Примеры команд SHELL:

1) `(mv f1.c f2.c; cp f2.c f3.c) &`
2) `sort -n < source | grep 'xyz'| cat > result`
3) `( ls -R dir1 | grep 'mydir'; (pwd; cd .. ; pwd) | sort) | wc`

Для выполнения задания по моделированию интерпретатора команд можно ограничить синтаксис входного языка; описать полученный вариант в виде БНФ или синтаксической диаграммы.

Обязательно должны быть реализованы следующие конструкции:

- конвейер
  pr1 | pr2 | ... | prN
  для произвольного N>=2;

можно считать, что аргументов у prI (1<=I<=N) нет (но возможна реализация с произвольным числом аргументов у каждого процесса)

- перенаправление ввода-вывода

<, >, >>
(в том числе для pr1 и prN в конвейере)
Примеры:
```
pr < data > res
pr1 | pr2 > res.txt
```

- запуск в фоновом режиме &
  (в том числе и для конвейеров)
  Примеры:
```
pr arg1 arg2 &
pr1 | pr2 | pr3 > res.all &
```

Должна быть реализована хотя бы одна из следующих операций:

1) pr1 ; pr2 ; ... ; prN
(последовательное выполнение команд prI - как если бы они были переданы интерпретатору по одной команде в строке)

2) pr1 && pr2
(выполнить pr1; в случае успеха выполнить pr2)

3) pr1 || pr2
(выполнить pr1; в случае неудачи выполнить pr2)

Приоритет операции '|' выше, чем приоритет операции ';' ,т.е. например, pr1;pr2|pr3 даст тот же эффект, что и pr1;(pr2|pr3). Однако, допустимо (pr1;pr2)|pr3, что приведет к конкатенации результатов работы pr1 и pr2, которые будут переданы процессу pr3 как входные данные. Операции '&' и ';' имеют равный приоритет.
Самый низкий приоритет у '||' и '&&'. Наивысший приоритет у '<', '>>', '>'.

Желательно реализовать хотя бы одну из следующих возможностей:

a) возможность исполнения командных файлов, содержащих команды из множества команд, определяемых вариантом.

b) возможность выполнения команды cmd путем передачи ее в качестве параметра при вызове
my_shell:
```
=> my_shell cmd
```

c) выполнение команд, указанных в круглых скобках в рамках дочернего процесса my_shell
Пример: `(ls|wc)|cat`

Команды, обязательные для реализации:
(«реализация» внешней команды означает написание соответствующей программы на языке Си; имя исполняемого файла, полученного в результате компиляции данной программы должно совпадать с именем команды; рекомендуется все команды для my_shell разместить в одном подкаталоге)

1) cd <каталог> - изменить текущий каталог (внутренняя команда)
2) pwd - напечатать имя текущего каталога
3) ls - вывести список файлов текущего каталога

Возможен (т.е. реализовывать необязательно) список файлов-аргументов: если аргумент является каталогом, то печатается список файлов этого каталога, в противном случае печатается сам аргумент.
Возможны флаги:
-R – вывести имена всех файлов текущего каталога, а также файлов, содержащихся во вложенных подкаталогах
-l – показать атрибуты: тип, права доступа, имя владельца, размер
-g – показать имя группы владельца

Если файл является символической ссылкой, желательно указывать также и имя файла, на который он ссылается, например mydir/f1→ /user1/f2 → /user2/subdir/f3. Здесь mydir/f1 и /user1/f2 - символические ссылки, /user2/subdir/f3 - обычный файл.

Далее перечисляются группы команд. Хотя бы одна команда из каждой группы должна быть реализована.

I.

1) mv old_file new_file (аргументы могут быть каталогами)
2) cp file copy_file
3) ln [-s] original_link new_link

II.

1) wc filename

результат: filename строк слов символов (возможен список имен файлов; в этом случае подобная информация выдается о каждом файле)

2) grep substring filename

результат: строки файла filename, содержащие substring как подстроку (возможен флаг -v ; в этом случае результат — это строки, которые не содержат substring как подстроку)

3) cmp filename1 filename2
(сравнение содержимого двух файлов)

результат: информация о первом различии

Пример: `filename1 differs from filename2: line 5 char 36`

4) sort filename
(сортировка строк файла в соответствии с кодировкой ASCII)

возможны флаги:

-r - обратный порядок
-f - не различать большие и малые буквы
-n - числовой порядок
+n - начать сортировку с (n+1)-ой строки

III. 1) cat filenames

возможен флаг:

-n – с нумерацией строк (если файлов несколько, то нумерация сквозная)

2) tail filename
(вывод 10 последних строк файла)

возможны флаги:

-n n последних строк
+n – с n-ой строки и до конца файла

3) od filename
(вывод содержимого файла по 10 символов в строке с указанием номера первого символа в каждой десятке)

Пример:
```
000001 a b c d \n e f g h i
       000011 j k \t l m n
```

возможен флаг:

-b – с указанием восьмеричных кодов символов

Замечания:

а) предполагается, что шаблоны в именах не используются;

б) очень желательно проверять синтаксис вводимых команд (например, методом рекурсивного спуска); в крайнем случае можно считать, что на вход интерпретатору подаются синтаксически верные команды; однако, интерпретатор должен выдавать диагностику в том случае, если он не может выполнить какую-либо команду, и причину отказа. Например, если во время выполнения команды надо открыть файл, которого нет в файловой системе; не удалось создать канал между процессами и т.п.

в) вариант задания определяется студентом самостоятельно путем выбора конкретного подмножества команд. Более содержательный вариант входного языка допускает:

- расширения, сформулированные с использованием слова "возможно"
- отказ от ограничений на использование шаблонов имен и предположения о синтаксической правильности команд
- реализация синтаксиса SHELL без сокращений