Вопросы (1 лабораторная работа UNIX)
По ps -al:
1. Коротко назвать назначение информации которую мы видим на экране. Перечислить какую информацию система выдает по процессам, прямо по полям.

Ответ:

F - Флаги (0 был fork, exec; 1 - был fork не было exec; 4 - суперюзер)

S - состояние, в котором на данный момент находится процесс.

UID - идентификатор пользователя.

PID - идентификатор процесса.

PPID - идентификатор родительского процесса.

C - процент использования процесса системой.

PRI - системная составляющая приоритета процесса, выше число - ниже приоритет.

NI - пользовательская составляющая приоритета процесса.

ADDR - не спрашивает

SZ - размер в физических страницах основного образа процесса.

WCHAN - это адрес в ядре, где процесс спал (если он действительно спит).

TTY - Терминал с которым связан данный процесс.

TIME - процессоров время, занятое этим процессом.

CMD - команда, запустившая данный процесс «с некоторыми опциями выводит и каталог откуда процесс был запущен».

q: Почему у процесса в столбце tty стоит знак вопроса(?) ?
a: Этот процесс - демон. У демонов нет терминала.

Тут может быть вопрос может ли система сама выполнить exec? НЕТ, поскольку в exec в качестве параметра передаётся исполняемый файл, а система что передаст.

2. Флаги, какие флаги определены в системе? Что они означают?

Ответ:

0 - был fork() и был exec()

1 - был fork(), но не было exec()

4 - использованы привелегии суперюзера

3. Что значит права суперюзера?

Ответ:

Суперюзер - это привилегированный пользователь, имеющий доступ к структурам ядра и привилегированным командам.

• Что такое exec?
Системный вызов exec() создает таблицу страниц для адресного пространства программы, переданной ему в качестве параметра, а затем заменяет старый адрес новой таблицы страниц.

4. Почему для системы важно, после форка был экзек или после форка экзека не было?

Ответ:

Когда выполняется системный вызов fork создаётся ещё один процесс, называемый потомком. Для процесса потомка создаются собственные карты трансляции адресов (таблицы страниц), но они ссылаются на адресное пространство процесса-предка (на страницы предка). При этом для страниц адресного пространства предка права доступа меняются на only- read и устанавливается флаг – copy-on-write. Если или предок или потомок попытаются изменить страницу, возникнет исключение по правам доступа. Выполняя это исключение супервизор обнаружит флаг copy-on-write и

создаст копию страницы в адресном пространстве того процесса, который пытался ее изменить. Таким образом код процесса-предка не копируется полностью, а создаются только копии страниц, которые редактируются.
После перехода на новое адресное пространство предку возвращаются права доступа read-write для сегментов данных и стека. exec заменяет адресное пространство потомка на пространство программы, переданной в exec и процессы перестают разделять адресное пространство.

5. Состояния, перечислить состояния, их смысл, что они означают?

Ответ:

D uninterruptible sleep - блокировка на вводе/выводе
R running or runnable (on run queue) - выполняется
S interruptible sleep (waiting for an event to complete) - заблокирован/ожидает
T stopped by job control signal - процесс остановлен
t stopped by debugger during the tracing - остановлен отладчиком
W paging (not valid since the 2.6.xx kernel) - процесс в свопе
X dead (should never be seen) - умер
Z defunct ("zombie") process, terminated but not reaped by its parent - зомби

6. Почему блокировку на вводе/выводе нельзя прервать (надо исходить из собственной практики программирования)?

Ответ:

Потому что это многоэтапный процесс в системе, прервать эту последовательность действий нельзя, потому что непонятно что и когда следует сохранять.

7. Кто такой зомби?

Ответ:

Это когда процесс потомок завершил свою работу раньше процесса предка и у него (потомка) отбираюся все ресурсы, кроме своего идентификатора, который нужен для того, чтобы процесс предок мог получить статус завершения потомка (строки в таблице процессов). И теперь если предок вызовет wait, то краха не будет и всё продолжит нормально работать.

8. Почему у родителя флаг 0, а у сына 1.

Ответ:

Потому что мы запустили новую программу из командной строки (это был fork() и exec()), а у чайлда не было exec().
(потому что мы запустили новую программу) (мы вызвали программу из командной строки, был предок мы вызвали новую программу поэтому был fork и exec) (для нашей программы предок это терминал)

9. Какие процессы называются процессами-сиротами? Какие действия выполняются в системе? Ну это уже ближе ко второй лабораторной работе.

Ответ:

Сироты - процессы, чей предок завершился раньше. В системе выполняется «усыновление» прародителем всех процессов - initом для восстановления процессов-сирот в иерархии процессов. Главная работа ядра UNIX при этом заключается в переустановлении PPID у "процессов-сыновей". Он становится для них равным 1. Кроме того, всем процессам - членам группы рассылается соответсвующий сигнал.

10. Изменить приоритет процесса, почему без прав суперюзера мы можем только понизить приоритет процесса?

Ответ:

Потому что тогда в системе будет хаос, если каждый пользовательский процесс сможет повышать свой приоритет, то будет происходить постоянный перехват в планировщике и все процессы будут постоянно откладываться.
(повышение приоритета означает - постановка в начало очереди)

11. Что показывает информация wchan?
Ответ:
Это адрес в ядре, где процесс спал (если он действительно спит).

По ls -a (ls - l)
1. Первая буква, 9 символов что они означают?
Ответ:
Первый символ означает:
d - файл является каталогом.
b - файл является специальным блочным файлом.
c - файл является специальным символьным файлом.
p - файл является именованным каналом.
Остальные 9 символов делятся на три группы по три символа: права доступа владельца, других пользователей из его группы, всех прочих пользователей. Внутри каждой группы используются три символа, обозначающие права на чтение, запись и выполнение файла соответственно. Для каталога под правом на выполнение подразумевается право на просмотр в поисках требуемого файла.

2. Столбик чисел, что он показывает?
Ответ:
Кол-во ссылок на файл???

3. Надо создать жесткую ссылку, гибкую ссылку, канал. (Про все спрашивает).
Ответ:
ln файл имя_ссылки - жесткая ссылка
При создании имя файла заносится в каталог и для адресации файла на диске создается указатель inode (I – Node номер), который содержит информацию о файле. После выполнения команды ln в каталог заносится новая запись, указывающая на inode существующего файла. Следовательно links имеют один и тот же inode

хард - еще одно равноправное имя файла

софт - специальный файл содержащий символьную строку, являющуюся путем к файлу

soft_link (ссылка на файл, а именно это специальный файл, в котором только одна строка содержащая путь по которому нужно перейти при обращении к этой ссылке, soft__link может ссылаться на несуществующие файлы)

q: Когда файл может быть удален из системы?
a: Когда на него нет hard_link.

4. Если на все отвечаем, передать сообщение между консолями.
Ответ:
Набираем в терминале:
mknod pipe p
echo Hello > pipe
Переходи на новое окно в терминале:
tee < pipe
видим, что высветилось Hello

5. Может попросить показать номер INODa.
Ответ:
ls -i

6. Изменить приоритет процесса:]
Ответ:
Для этого запускаем прогу parent и child, где они в бесконечном цикле.
Переходим на новое окно терминала. Там набираем ps -al. пишем
renice 10 id_процесса(id дочернего процесса из ранее запущенных) .
Тут 10 - новое значение пользовательской составляющей приоритета (NI).
Снова набираем ps -al и видим что в компоненте NI стоит теперь не 0 а 10.
Изменять без root (а его у нас на тех компах нет) можно только в большую сторону. Т.е. теперь можно написать только что-то типо:
renice 12 id_процесса

Потомок не знает, что вейт в предке есть. (Не знаю что это и зачем, но НЮ попросила это записать).

• Зачем нужна иерархия процессов в юникс?
Чтобы не было лавинного "сброса" информации (статусов завершения процессов).
Чем выше процесс по иерархии, тем меньше информации поступает в него от других процессов. [Можно при ответе изображать руками что-то вроде конуса - плюс в карму]
s - состояние процесса. Основные:
R - running (выполнение)
S - state(заблокирован)
Z - зомби

• Что такое exec?
Чаще всего нет смысла в выполнении двух одинаковых процессов и потомок сразу выполняет системный вызов exec(), параметрами которого является имя исполняемого файла и, если нужно, параметры, которые будут переданы этой программе. Говорят, что системный вызов exec() создает низкоуровневый процесс: создаются таблицы страниц для адресного пространства программы, указанной в exec(), но программа на выполнение не запускается, так как это не полноценный процесс, имеющий идентификатор и дескриптор. Системный вызов exec() создает таблицу страниц для адресного пространства программы, переданной ему в качестве параметра, а затем заменяет старый адрес новой таблицы страниц.
Если попросит коротко, то exec переводит процесс на новое адрессное пространство.