

ОС. Семинар 3

28.09.24г.

К лаб/р. 1 Unix:

1. Симлинк - специальный файл, содержащий путь к файлу.
 2. Имен. программного файла - спец. файл, буферизирована FIFO.
 3. Жёсткая ссылка (hardlink) - ещё одно правоуправление на файл.
- ! Не называть файлы "hardlink", "symbolic", "pipe" и т.д.

Системный вызов FORK. —

создаёт новый процесс - томашок, который является копией процесса - предка, т.е. наследует от предка дескрипторы микроЭОХ файлов, сигнальную маску и перененное окружение.



Для процесса - томашка создаётся собственное защищённое виртуальное адресное пространство - таблицы страниц.

В классической Unix были таблицы симлинков - картины прокл. адресов.

Ноу предка ПОЛНОСТЬЮ копировалася в адресное пр-во томашка.



В старых системах. В рез-те: в системе могли существовать несколько копий одной программы.

В современных системах: дескрипторы страниц в таблицах страниц процесса-памяти сначала лежат на страницах адр. пр-ва процесса-предка.

Три этапа: для страницы адр-ва пр-ва предка права доступа меняются с r/w на only read. и удаляется "copy-on-write" (копирование при записи).

И.е., когда процесс предка/потомок пытается редактировать страницу адр. пр-ва предка, произойдёт исключение (read only).

Вынужденное исключение, система обнаруживает оно как "copy-on-write" и создаёт копию редактируемой страницы в адресном пр-ве того процесса, который пытается её изменить.



В рез-те: создаются копии отдельных (изменённых) страниц.

Также существует в синтаксе существуют, пока процесс-потомок не возвратил статус вызов **exit(..)** или сис. вызов **exec(..)**.

→ завершаем процесс

переводим процесс на новое addr.пр-во
дел запуска на выполнение программы,
передаваемой в качестве параметра.

Лабораторная работа № UNIX.

I. Чемаре программист.

① Создать не менее 2^x потомков, необходимо создать их серийки.
В конце child добавить sleep().

Выводы: PID, PPID, GID.

```
if (childpid == φ) {  
    printf ("%d,%d,...", getpid(), getppid(), getgpr())  
    sleep(2)  
} else —  
    printf ("....", getpid(), childpid, getgpr())
```

Поскольку бенчм. то fork'у (бенке) выполн.

- след. обр:
- потомок получ. 0
 - предок - PID потомка =>
 - -1 - неудача.

=> в конце прокла пишем еще переменной, т.е. **childpid**.



Объекты массивов (pid +) для РН потоков, чтобы вызвать вселину в цикле for.

pid + childpid; (когда они где int) Это связано с менеджером: для каждого типа объекта выделяется свой тип для осуществления дополнительного контроля со стороны типов.

Контроль автоматически возможен классификации.

2 В parent'e вызываются системные вызовы wait(), убираем sleep()
wait() - после цикла for() !!!

Два типа wait(): wait(), wait-pid(). См. MAN

Актуализируемые списки завершающихся потоков.

1^{ый} поток - завершается нормально, а

2^{ой} необходимо убить с помощью kill

↳ подвесить на бесконечн. while, pause().

3 В потоках вызываются системные вызовы exec(). См. MAN Example exec, execve

Минимум 2 потока.

Передавать свое программное на См.

Обязательно: применение ввода!!!

1^{ый} процесс должен блокироваться в ожидании ввода (⇒)

⇒ таким передаете флаги процессу.

④ Взаимодействие процессов комманд с процессами другом через неизменяемой программной комм.

PIPE()

Все системное вводы проверять на - 1.

Parent - пишет

Child - читает

в примере
MAN

Оба конца пишут

Parent читает

В ладе

AAAA

Сообщение г.б. графной формат: BBBB... BBBB.