МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени Н.Э. БАУМАНА

Учебное пособие

Учебно-методический комплект

по курсу "Системное программирование"
Конспект лекций

МГТУ имени Н.Э. Баумана

Лекция №1

Гудошников И.В
434-12М

1) ОС - совокупность программ, которые решают ряд задач:

многопользовательность

многозадачность

управл. процессами / понятью

взаимодействие с человеком и периферией

обеспечение безопасности (негативное

влияние процессов друг на друга и на

данные)

Классификация задач:

По архитектуре (x32, x64),

По типу ядра (

2) Дисковая ОС (однозадачные) - Выполняет

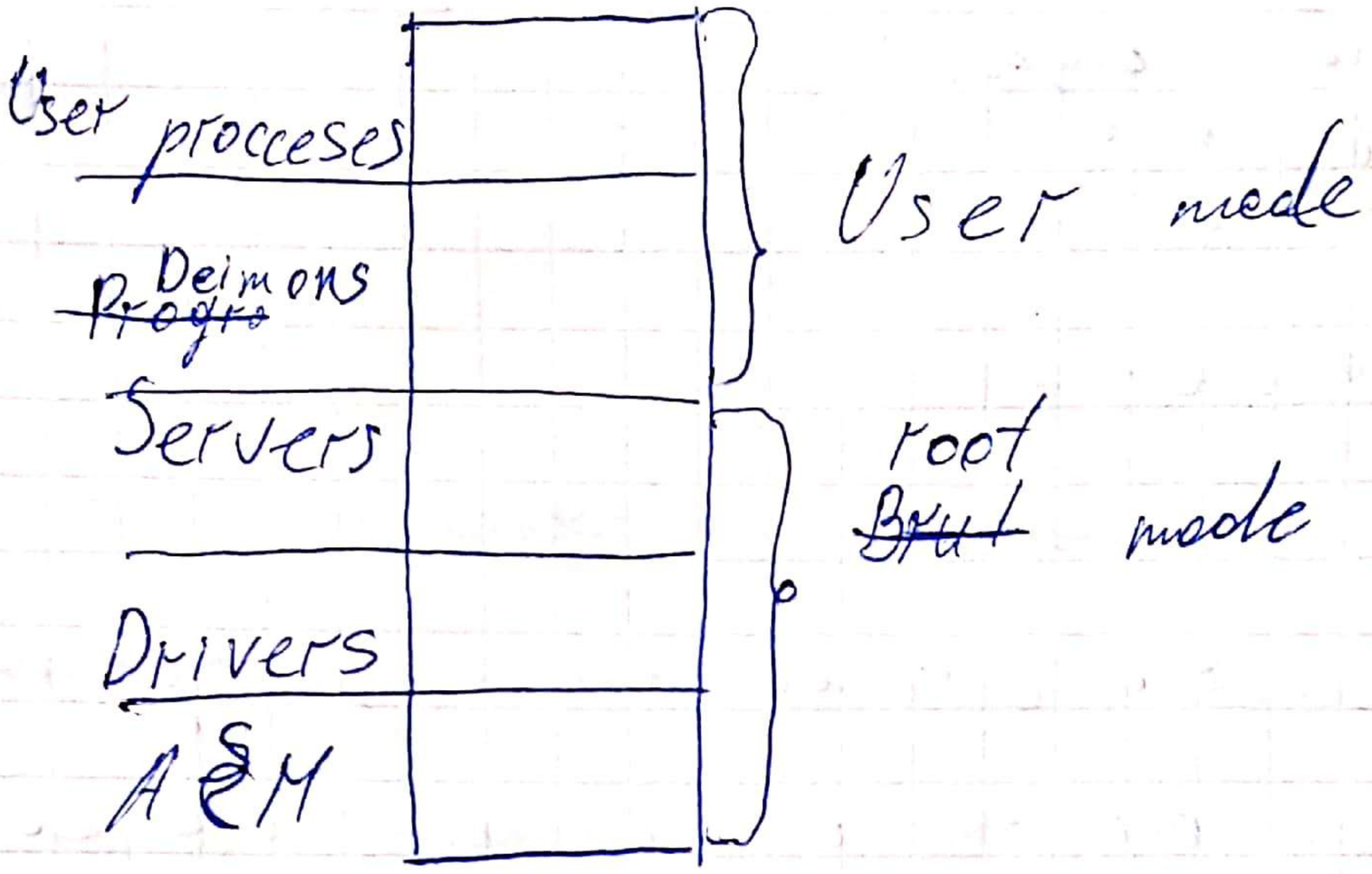
задачи должна явно передать управ. ОС

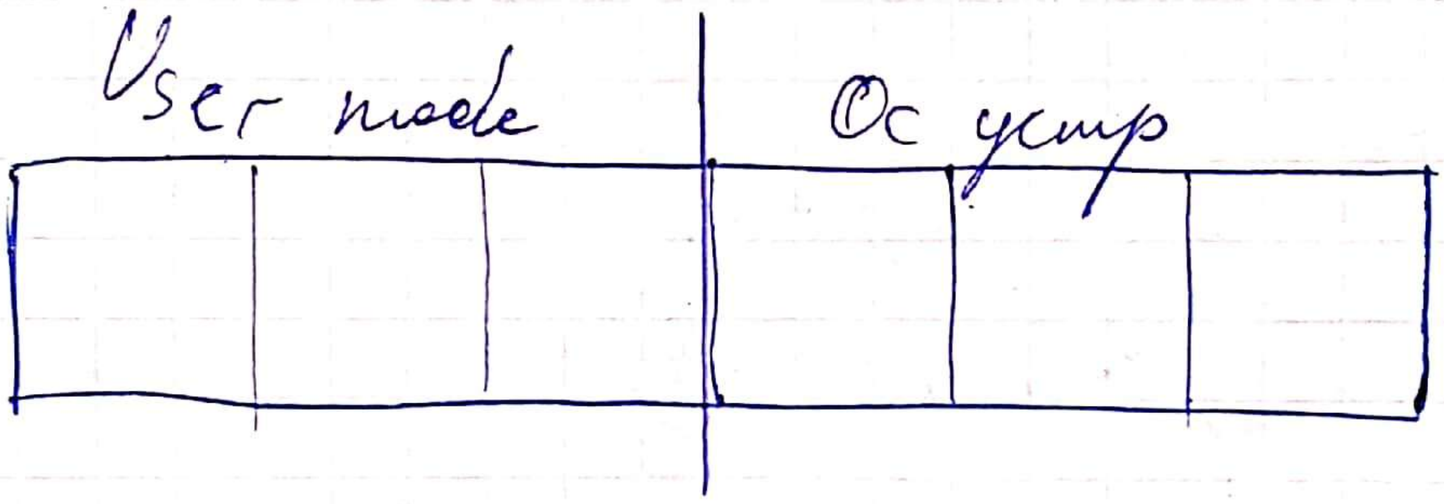
3) ОС общего назначения (чувствительнее)

4) ОС реального времени (гарантирует

выполнение ~~задач~~ задачи к определенному

моменту времени после её старта)





ОС состоит из следующего

ASM - ассемблерно - зависимая прослойка на языке Assembler

Drivers - набор программ отвеч. за работу с периферией (как правило сводится к работе с портами ввода вывода буферы и регистры устройств)

Servers - программы, отвечающие на запросы других программ, выполняют сервисные функции: process manager, file man.,

Deimons - совокупность процессов, выполняют сервисные функции. - Scheduler не имеет интерактивного интерфейса.

User processes - программа, выполн в пользовательском пространстве по запросу оператора

Режимы работы системы с памятью

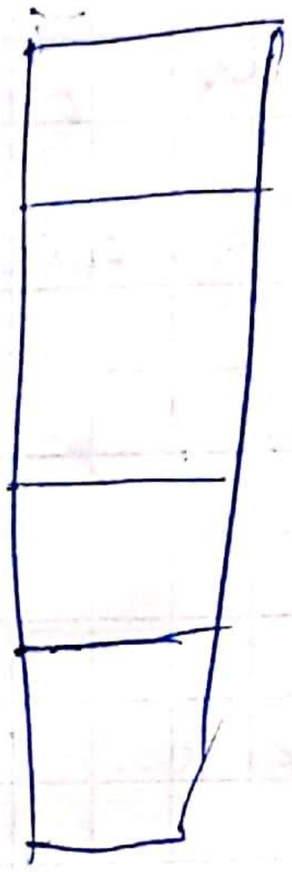
Kernel mode Режим ядра - в этом рем. ОС доступна вся память. Как правило первые 3 уровня

Режим пользователя - программам доступен ограниченный объем памяти. Недоступны адреса ОС и устройств. Как правило 2 последних уровня.

Stack segment

Data segment

Code segment



Процесс - программа в момент её выпол. ОС имеет 2 сущности для решения задач управления процессом: 1) образ памяти процессов (см. рис ↑) :

Code segment - содержит исполняемый ~~код~~ программы

data seg. - память доступная польз. и программисту

stack seg. - служебная память, необх. для работы программ.

2) таблица процессов - содержит по 1 записи для каждого процесса. Хранит все служебные параметры, указывающие процесса.

среди них:
process id, приоритет, статус,
стартовая команда, расположение образа
памяти и т.д.

Семинар №1

Кто?
откуда?
Где находится?
ОС / Linux
операционная

Linux debian понятнее Ubuntu/Debian!
без GUI (терминал)
Вендоров Solaris +

Установливает (любой
Backup на флешке!

512 байт - минимум размер с HDD

CHS
цилиндр head sector
8 бит 8 8

Record
Master Boot ~~Report~~
↓
512 байт

При загрузке запрашивается

Primary Boot Loader 446

Partition location table 64 байта

byte [3] - start

byte [3] - stop

int begin

int end

byte - active

byte - file system

Magic number 0xAA - 2 байта

1 2 Файловая система и пользователь -

2-3 Управление процессами

4 Bash

ssh (pu-Hu)

NAT

Bridge 1

man - manual

whereis - путь к утилите

which -

mc - total comand

pwd - место

whoami - пользователь

who

cd ~ домашняя директория

ls

ls -a

ls -l

history - все команды

ls -al

ctrl-D - убить все процессы

man → less more

/usr/share/man

.bashrc - run a command

.bash_history - логи команд пользователя

/var/log - все логи

-l - long

битовая маска доступа

тип файла	xxx	xxx	xxx	xxx
	rwx	rwx	rwx	15 бит
	user	group	other	suid sgid stick

доступ mode

set user id - suid

set group id - sgid

getty

P|O

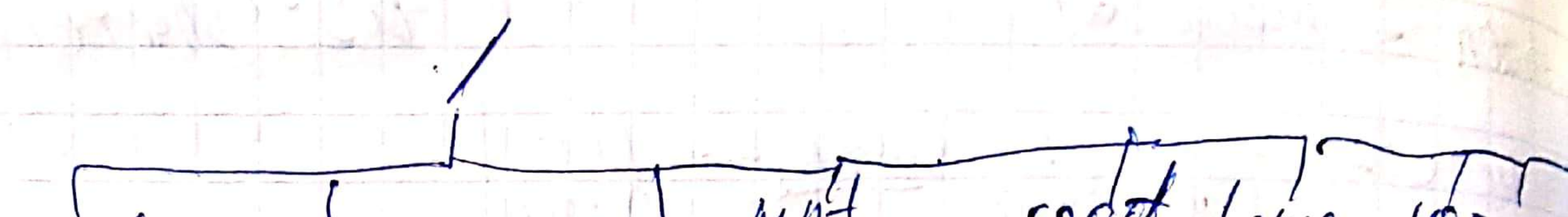
DPID

UID

GID

EUID ← SUID

EGID ← SGID



Создание user:

1. Создать запись в passwd и shadow.
2. Создать home директорию
3. Изменить владельца home директории и всех скрытых файлов для user'а

Ch~~mod~~own - изм. пользов

adduser }

ver add }

userdell

chmod - переназначение прав

touch - создание файла

nano - текст. редактор

echo $? - как завершилось последняя команда

0 - true

"-" regular file

"/" если в каталоге

id

␣ пробел - разделяет слова (команды)

; переход на новую строку (Enter) L

& фоновый

| pipe

&& pc1 && pc2 (pc2 если pc1 „0")

|| pc1 || pc2 (pc2 если pc1 „не 0")

~ - home

$ - переменная не echo, а bash

\ esc

" " ' ' в " $ не теряет значение

\ \ способ записать вложенную кучу

< > перенаправ потоков ввода-вывода

>> не затирать, а внести в конец

*, ?

Переменные

виды

var=5 – переменная в среде

export var - export в как переменная среды

env / ~~enf~~ - все используемые переменные

unset var – удалить переменную

Спец. перем

? @ # !

? результат прошлой команды

@ сами параметры

кол-во ~~процессов~~ параметров скрипта

!

Позиционные параметры

./myscript one two

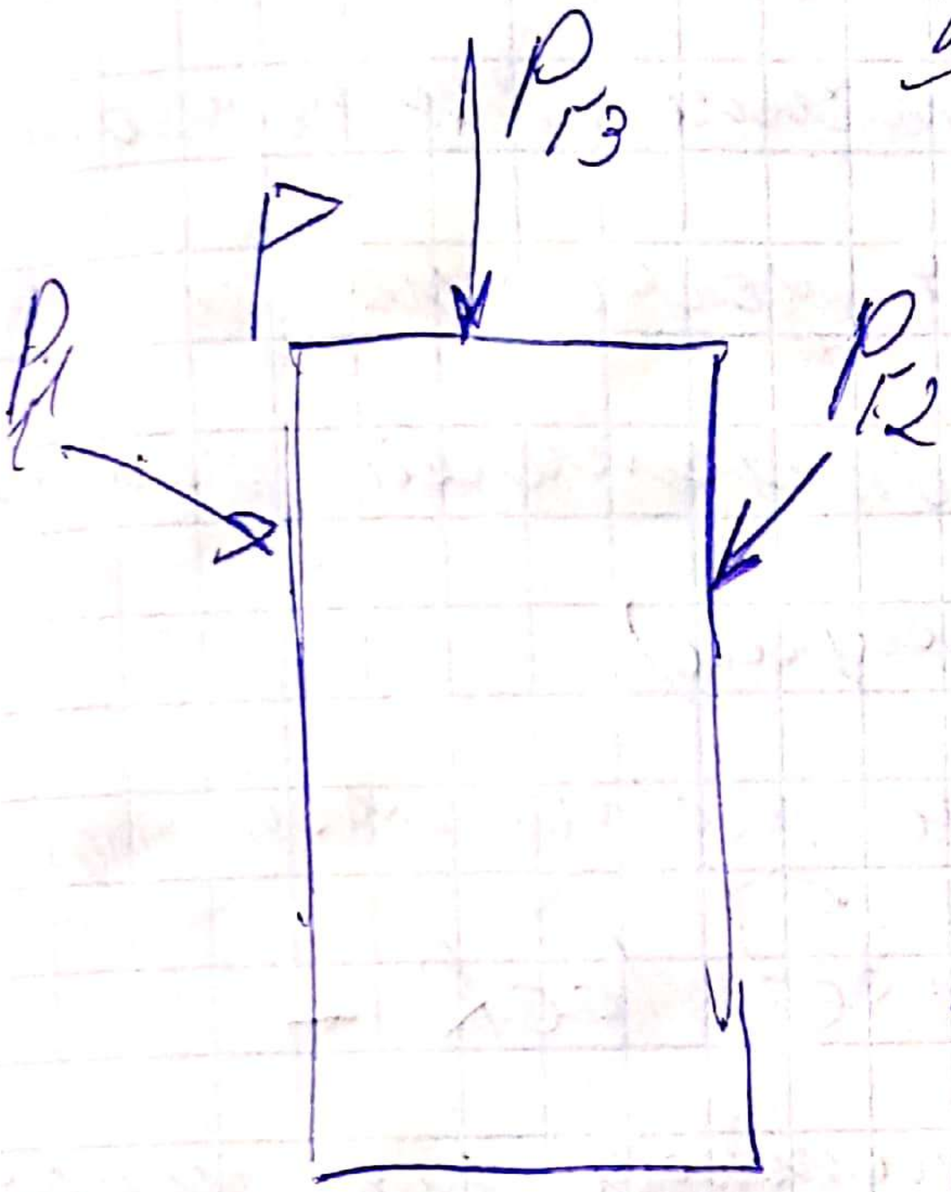
myscript

echo $0 myscript

$1 one

$2 two

Последовательность обработки от лекция !!!



Инверсия прерываний

Это негативное последствие совместного решения процессов к разделяемому ресурсу

В простейшем случае процесс при получении отказа на доступ к ресурсу:

1) Продолжить в бесконечных запрос доступа
2) Перевести себя в ожидание

В конечном итоге это приводит к тому, если в состоянии ожидания стоит высокоприоритетный процесс, между процессов понижает его приоритет

Описанная ситуация может возникнуть в случае, если планировщик процессу P_1 после завершения работы с разделяемым не успел его освободить с помощью ПМ (Process Manager),

Consconsumer. proy. Pr2 будет стоять в очереди и ждать своих блокировок от Pr1. По этой же причине может быть получена ~~ошибочная~~ deadlock - раздел. рес. (модиф. указателей доступа после завершения операции)

Существует понятие неделимой операции

Например: test ~~and~~ set lock - гарантированно короткая неделимая проверка и операции и установки флага

Лекция

Режимы адресации в языках Assembler

Существует несколько режимов адресации в командах ASM, в общем случае их можно увидеть по:

- факт результирующего кода
- времени выполнения и конкретной команде.

Режимы:

1) Операнд - константа

Данные в теле команды

2) Самый быстрый и самый удобный

2) Операнд - регистр

операнд в регистре, в теле команды указатель на регистр.

самый короткий. Длина кода сокр.

до длины адресного пространства регистров.

Увеличение скорости.

3) Прямая адресация

Данные в ОЗУ, адрес в теле команды.

Длина уменьшается до длины адресного пространства системы, скорость увеличивается

до времени на запрос ОЗУ

4) Регистр адресации

данные в ОЗУ, адрес в регистре, № регистра в теле команды

Длина кода 2м, время работы увелич.

5) Косвенно-регистровая адресация

Рассмотрим в современных системах адр. (статически сегментной ~~регистр~~). Адрес базы в регистре, смещение в команде, сумма в сумме

На момент ассемблирования исх. код неизвестно имя. Все перемещаемые данные попадут после перем. комп. сегм. метки и вносятся в отдельную структуру в которой имя. операционной среды в момент загрузки и вычисляется команда Структура - RelocationTable, её формат зависит от операционной среды и от формата исполняемого файла.

Лекция

Структуры данных

Массив - самая простая стр. данных.
Основное свойство - элементы памяти
располагаются всегда последовательно.
Можно исп. усложненные модификации
указателей

Связный список - стр. д., которая дает
возможность более гибкой работы со связями
элементов по сравн. с массивом (переход
от LAST к FIRST эл., операции вставки и
удаления перемещены)

Стек и очередь (LIFO / FIFO) - удобно для
решения задач межпроцессной комуни-
кации. (Т.к. разработчикам ОС удобнее
не нужно договариваться о понятии
„активный элемент")

Деревья - подвид п-связного списка (частный
случай - бинарные деревья - подвид 2х связ. списка)

Имеет логарифмическую сложность при решении задач сортировки

Куча - обод. в себе приоритетное двоичного дерева с точки зрения быстр. с. и массива с точки зрения пространственной сложности при решении задач сортировки

В этой стр. д. для связи родителей и потомков исп. не доп. адресное поле, а свойство бинарного дерева, кот. заключается в том, что левые и правые потомки вычисляются по формуле:

$j = 2i + 1 (+2)$; левый (правый) потомок

а родитель по формуле:

$i = SIZE/2$; где SIZE - Номер потомка

Семинар

Целевая сборка

all : ресурсы

[TAB] команда

build.sh >> log

make all 2> ./my prog 1 out log

make clean

Типы переменных в СИ:

int
char
void
double
float

(Lg) - с/-d как СИ

Лекция
Сложность алгоритмов

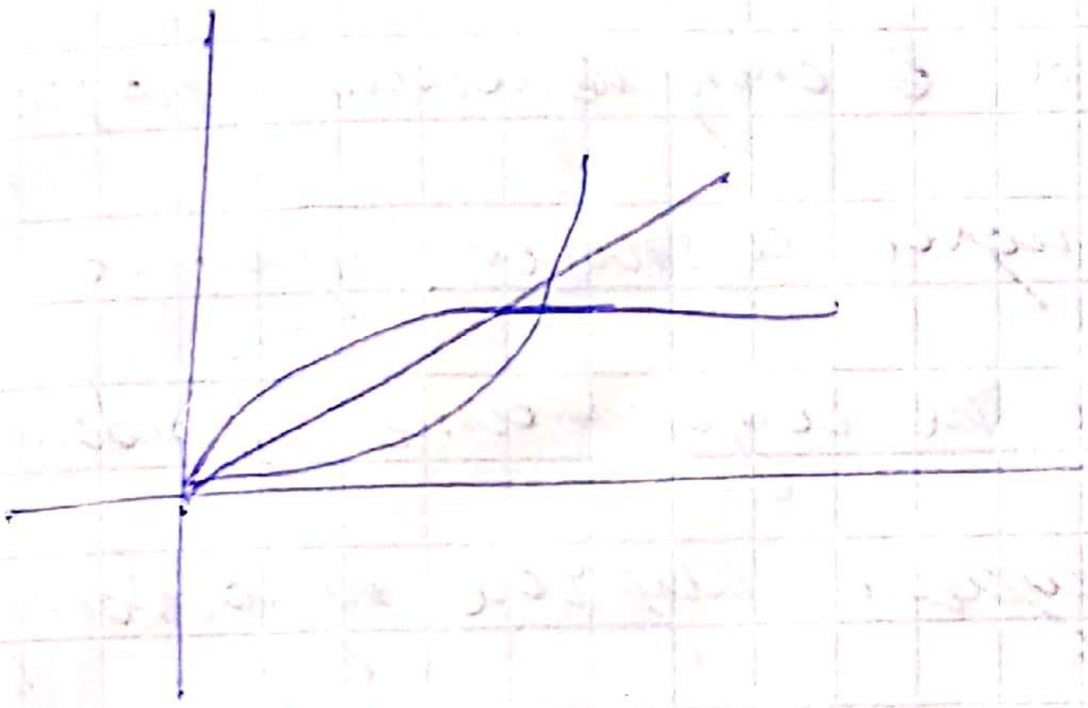
Задача анализа сложности возникла из-за необходимости оптимального использования ресурсов: (памяти) пространства и времени (вычислений).

Анализ вых. сложности алгоритма произв. путем разбиения его на элементарные операции и подсчета того сколько раз в зависимости от мощности входа. Эта операция будет повторена. Чем более детально разбиение, тем более точную оценку можно получить.

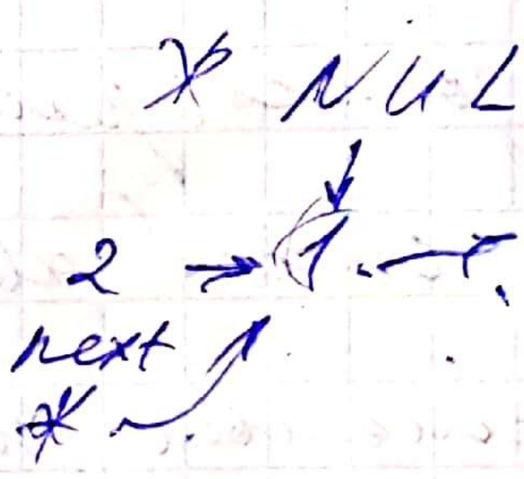
В терминальном случае - рассчитать кол-во секунд (тиков) на вып. (используя документацию на систему). В большинстве случаев к асимптотической оценки самого сложного вар. входа.

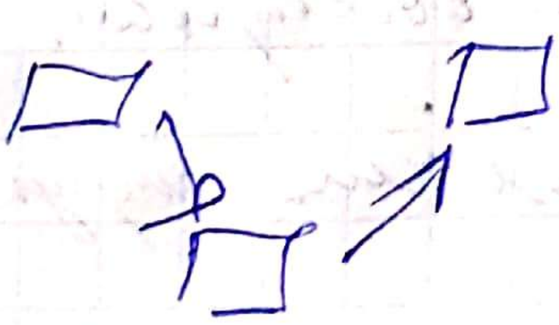
Вых. сложность - степень полинома, выраж. зависимость времени вып. алгоритма от входа. Важно понимать, что быстро растущие

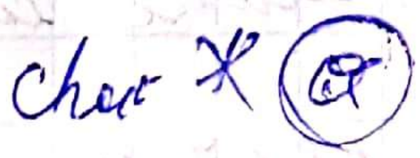
чая функция сложности показывает
лучший результат на маленьком входе,
он медленно растущую

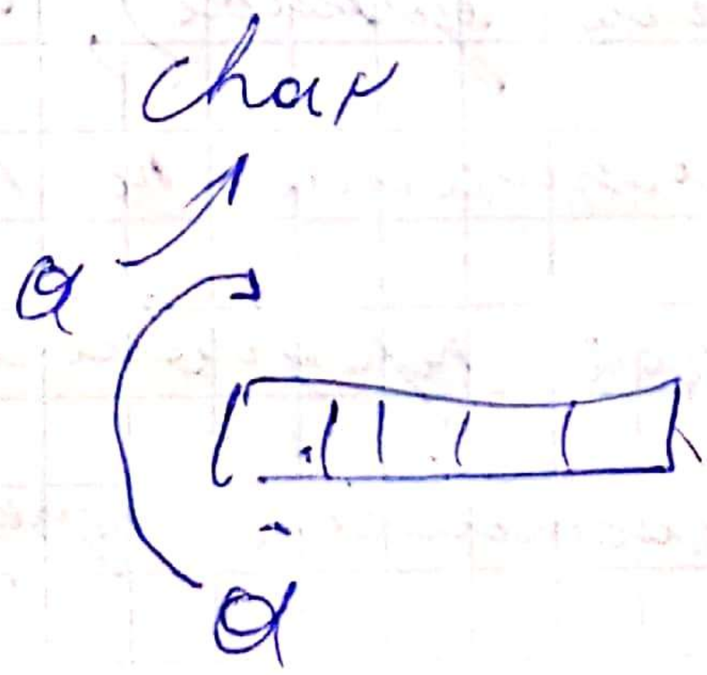


Пространственная сложность - асимпт.
оценка объема памяти, которую займет
алгоритм для решения задач.









Лекция

Межпроцессное взаимодействие

Сигналы - самый простой, самый быстрый, самый не универсальный способ межпроцессного взаимодействия. Заключается в отправлении процессу короткого идентификатора сигнала. Процесс может содержать или не содержать обработчик для сигналов. В последнем случае обработку выполняет системный обработчик.

Каналы - механизм, выделяющий область памяти для межпроцессной коммуникации (IPC) и выдающий к ней 2 fd: один на чтение, другой на запись (реализована очередь внутри pipe). Этот способ является самым простым из универсальных (позволяет передавать любые данные), а также самым быстрым. Ограничение не позволяет управлять структурой передаваемых данных, не контролирует целостность передаваемых сообщений, не имеет механизма защиты от коллизий при множест

вещом доступе. Каналы бывают: именованные и неименованные в зависимости от разделения выделяемой памяти: в файловом пространстве или в оперативной памяти сесть. System five interface defenitions (SVID) - не йтитуционая библиотека методов межпроцессной коммуникации импортированная в Linux, функциональность которых аналогична pipe'ам. Также имеет фун. методы, позволяющие контролировать целостность и структуру передаваемых сообщений - messages.

Сокеты - кроссплатформенное семейство методов межпроцессной коммуникации.

Парные сокеты - аналог неименованных каналов.

Сокеты файлов пространства имен - дополняют именованные каналы возможностью управления структурой передаваемых сообщений.

Сетевые сокеты - позволяют взаимодействовать удаленным процессам, используя сет.

TCP/IP с управлением структурой сообщения
и гарантией доставки