Makefile - Assignment

* Normal Assignment
  obj-m := hello.o

* Recursive Assignment
  obj-m = hello.o

* Conditional Assignment
  obj-m ?= hello.o

* Additive Assignment
  obj-m += hello.o

SRCS = main.c fun.c
SRCS += add.c

Module stacking

- Module loading sequence can be automated using modprobe tool.
- modprobe tool loads modules from /lib/modules/`uname -r`.
  The module dependency is read from modules.dep file.
  This file is built using depmod tool.

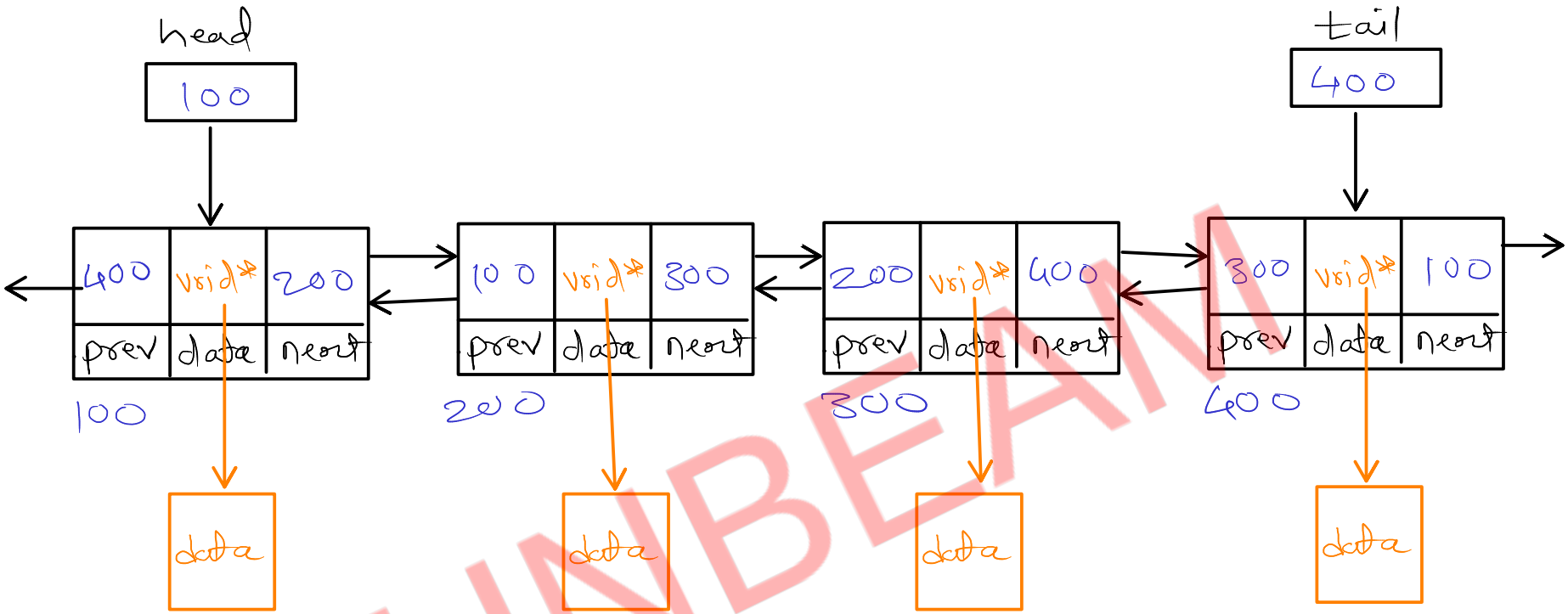
- Steps to use modprobe:
  terminal> sudo cp import.ko export.ko /lib/modules/`uname -r`/kernel
  terminal> sudo depmod

- Loading module using modprobe
  terminal> sudo modprobe import

- Unloading module using modprobe
  terminal> sudo modprobe import
  terminal> sudo modprobe export

- Modules can be loaded automatically while boot by making entry into /etc/modules.

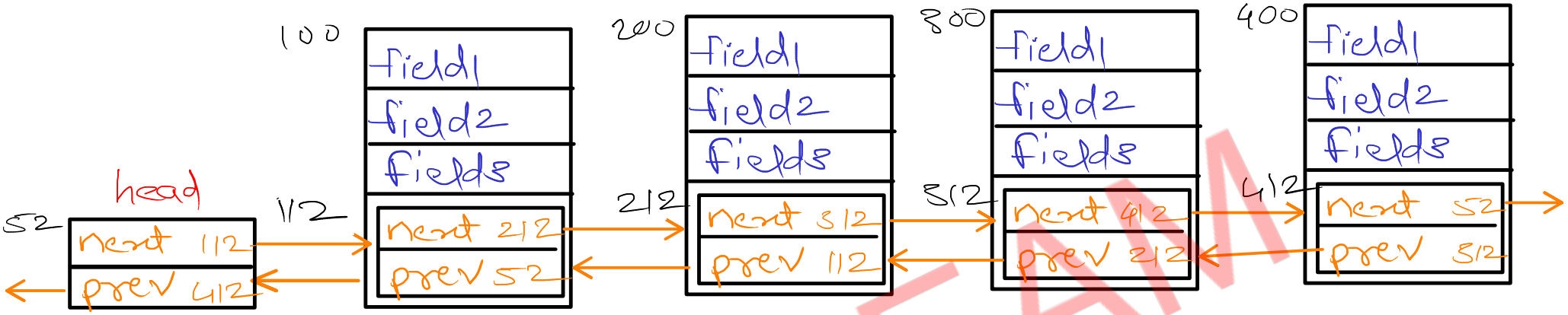
Kernel List



Generic Linked List

- same information is stored at multiple places (different data structures)
  - job queue / process list
  - ready queue
  - waiting queue

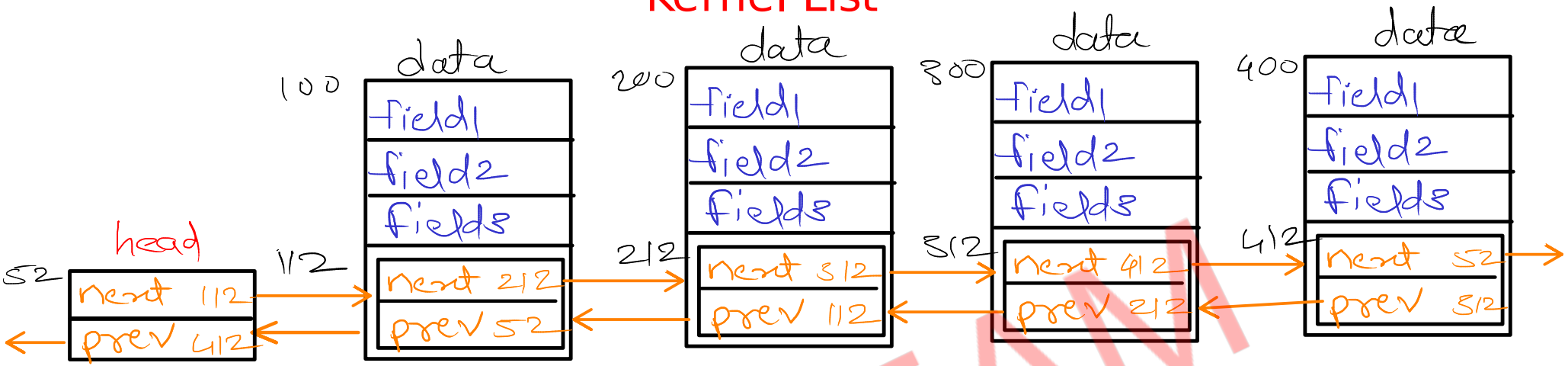
Kernel List




```
struct student{
    roll no ;
    name;
    marks;
};

struct node{
    struct student data;
    struct node *next;
    struct node *prev;
};
```

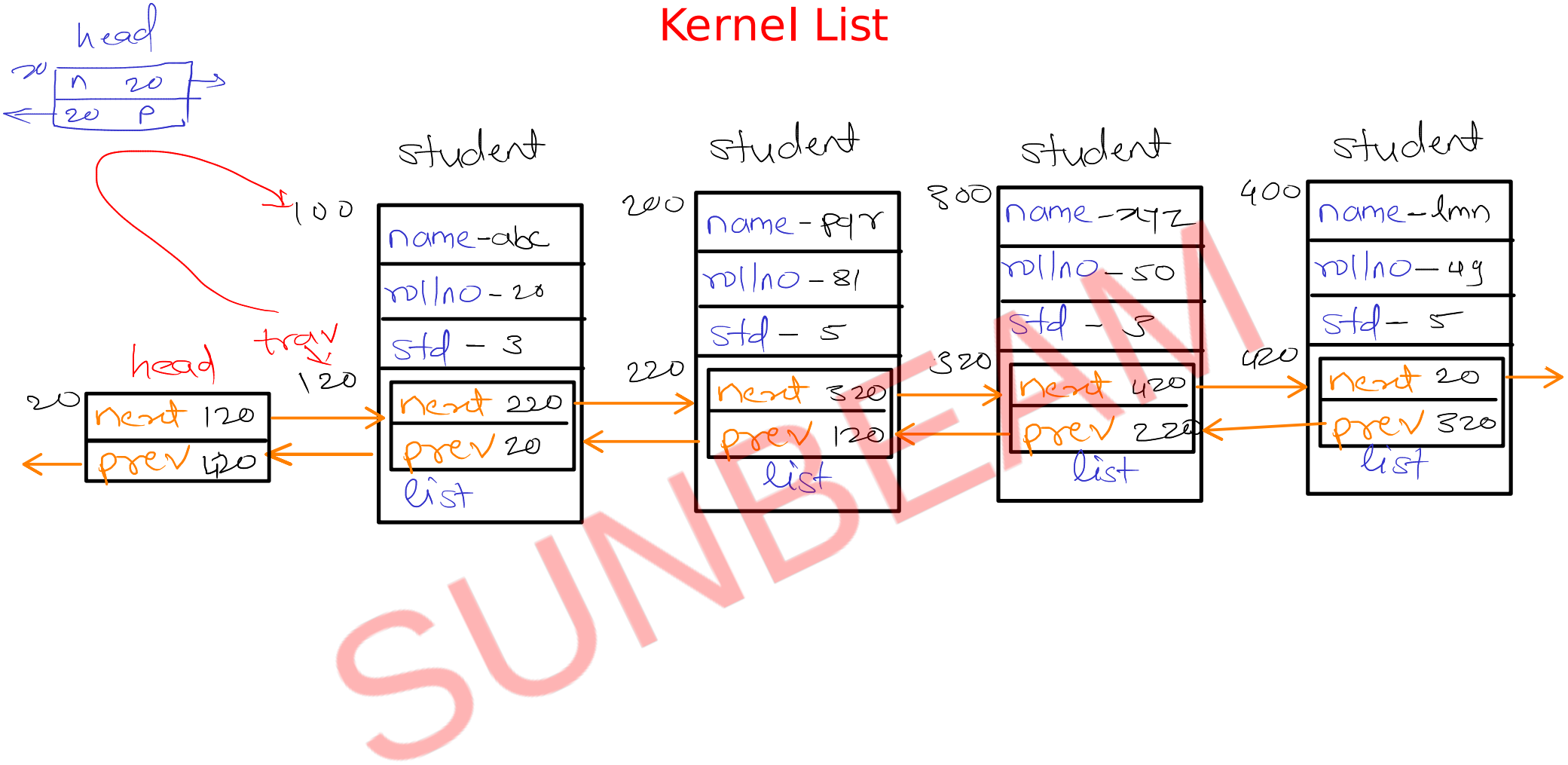
```
struct node{
    struct node *next;
    struct node *prev;
};
struct student{
    roll no ;
    name;
    marks;
    struct node n;
};
```

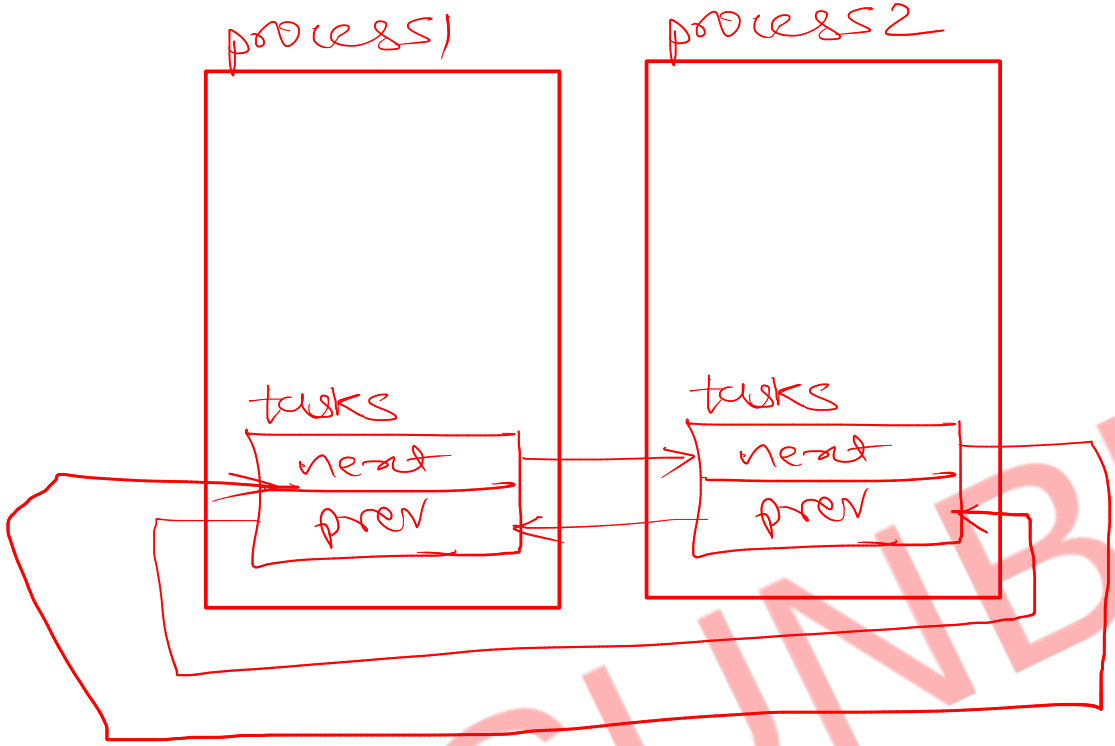
Kernel List




```
#include <linux/list.h>

struct list_head {
    struct list_head *next;
    struct list_head *prev;
};

struct data {
        field1;
        field2;
        fields;
    struct list_head list;
};

add_node(new, head);
add_node_tail(new, head);
del_node(ptr);
list_for_each_entry(----);

struct student {
    name;
    roll no;
    std;
    struct list_head list;
};
```

Kernel List





task_struct {

—

—

—

—

—

—

—

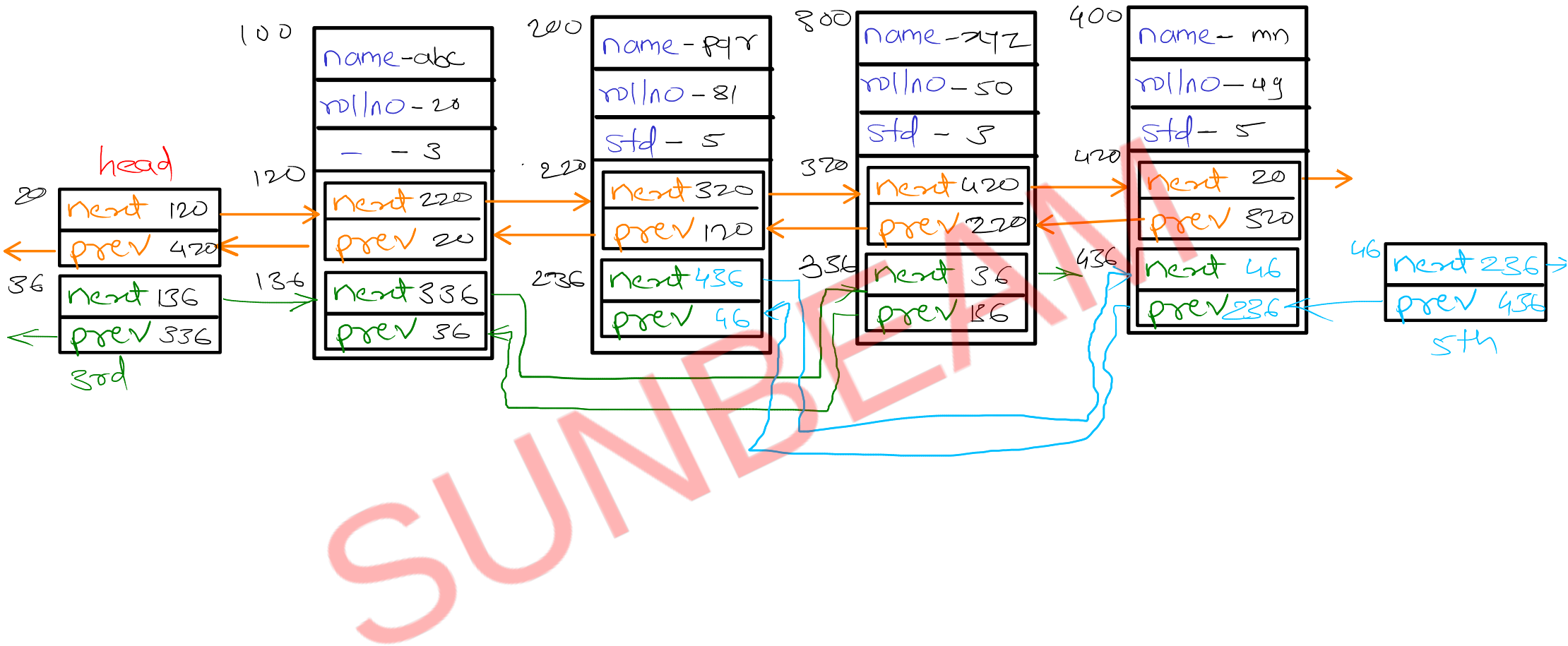
struct list_head tasks;

}

init_task.h

init_task

Kernel List



Kernel FIFO

- Kernel FIFO is circular queue using array.
- In kernel FIFO is represented by struct kfifo and is declared in
    <linux/kfifo.h>.

- struct __kfifo {
    unsigned int in; // like rear
    unsigned int out; // like front
    unsigned int mask; // to keep track of size
    unsigned int esize; // ele size
    void *data; // pointer to the (data) buffer
};

Kernel FIFO

- kfifo functions
  - int kfifo_alloc(struct kfifo *fifo, unsigned int size, gfp_t gfp_mask);
  - void kfifo_free(struct kfifo *fifo);

  - unsigned int kfifo_in(struct kfifo *fifo, const void *from,
    unsigned int len);
  - unsigned int kfifo_out(struct kfifo *fifo, void *to, unsigned int len);

  - unsigned int kfifo_size(struct kfifo *fifo);
  - unsigned int kfifo_len(struct kfifo *fifo);
  - unsigned int kfifo_avail(struct kfifo *fifo);

  - int kfifo_is_empty(struct kfifo *fifo);
  - int kfifo_is_full(struct kfifo *fifo);