Skapandet av liststrukturer och funktioner för listhantering

En stor del av administrationen i en realtidkärna består av listhanteringen. I detta inledande moment skall tre olika listor skapas. Dessa, plus deras element, har fördefinierade namn vilka några är beskrivna nedan. Det är viktigt att de nedan definierade strukturer etc. har det angivna namnet i ert program också.

Uppgift

- Skapa tre länkade listor.
- Skapa funktioner som kan stoppa in och plocka ut element ur dessa listor.
- Debugga och se att ni klarar olika storlekar på listor. "Från noll till många element."
- Ta ut listobjekt ur en lista och stoppa in i en annan.

Listornas funktion

Tre listor skall skapas: Timerlist, Waitinglist och Readylist. I detta initiala skede använder vi oss inte av alla delar i listelementen. (Beroende av listtyp kommer olika delar av strukturen att användas) Listorna är dessutom sorterade enligt var sin princip, men varför det skall vara så här beskrivs under föreläsningarna.
Den variabel (heltalet) som styr insättningen i listan kopieras över till elementets variabel *nTCnt.* Därefter sker insättningen m a p *nTCnt*.

Timerlist:

- Vid insättning i listan skickas ett heltal med vid funktionsanropet. Detta styr var objektet skall sorteras in. Insättning i listan sker så att elementet med lägst värde på *nTCnt* ligger först.
- Uttagning sker alltid framifrån, dvs vid "Head-elementet".

Waitinglist:

- En dubbelänkad lista, där insättning i listan sker så att elementet med lägst värde på TCB->Deadline ligger först.
- Uttag sker mha av en listelementpekare. *struct l_obj *pBlock;*

Readylist

- Elementet med lägst värde på TCB->Deadline ligger först.
- Uttagning sker alltid framifrån, , dvs vid "Head-elementet".

Listorna kan vara av typen:

```
// Generic list
typedef struct {
                listobj         *pHead;
                listobj         *pTail;
} list;
```

OBS. Det är med fördel man använder sig av en lista med huvud och svans! (andra lösningar kan också fungera)
Vid alla insättningsfunktioner så skickas en pekare till ett listobjekt med. Uttagningsfunktionerna returnerar en pekare till ett listobjekt. Om listan är tom returneras *NULL*.

List-elementen

```
struct  l_obj;          /* Forward declaration*/

/* Generic list item*/
typedef struct l_obj {
                TCB             *pTask;
                unsigned int    nTCnt;
                msg             *pMessage;
                struct l_obj    *pPrevious;
                struct l_obj    *pNext;
} listobj;
```

Samtliga listobjekt har dessutom en pekare till ett s k TCB-block. Detta definieras enligt:

```
// Task Control Block, TCB
typedef struct
{
                    void            (*PC)();
                    unsigned int    *SP;
                    unsigned int    Context[CONTEXT_SIZE];
                    unsigned int    StackSeg[STACK_SIZE];
                    unsigned int    DeadLine;
} TCB;


/* Generic list*/
typedef struct {
                     listobj        *pHead;
                     listobj        *pTail;
} list;
```

Som synes finns det vid detta stadiet av programmeringen flera variabler som inte används! Vi bryr oss inte om dem nu. Användningen av dessa kommer att inses lite senare i kursen.