

Plank's Jval and Dllist Libraries

CS350: Systems Programming

Instructor: Dr. Dorian Arnold
Computer Science, Emory University
Spring 2021

Jval: A generic “container” type

```
typedef union {  
    int i;  
    float f;  
    double d;  
    void *v;  
    char *s;  
    char c;  
    ...  
} Jval;
```

Always 8 bytes!

Jval Constructors

```
extern Jval new_jval_i(int);  
extern Jval new_jval_f(float);  
extern Jval new_jval_d(double);  
extern Jval new_jval_v(void *);  
extern Jval new_jval_s(char *);  
extern Jval new_jval_c(char );
```

for initializing with a particular type and value

Jval Accessors

```
extern int jval_i(Jval);  
extern float jval_f(Jval);  
extern double jval_d(Jval);  
extern void *jval_v(Jval);  
extern char *jval_s(Jval);  
extern char jval_c(Jval);  
...
```

for extracting a particular type and value

dllist: A generic doubly-linked list container

```
typedef struct dllist {  
    struct dllist *flink;  
    struct dllist *blink;  
    Jval val;  
} *Dllist;
```

Dllist Functions

```
extern Dllist new_dllist();
```

allocate and return new list

```
extern free_dllist(Dllist l);
```

destroy l, freeing all memory. List can be non-empty.

```
extern dll_append(Dllist l, Jval v);
```

insert v at end of list, l

```
extern dll_prepend(Dllist l, Jval v);
```

insert v at beginning of list, l

```
extern dll_delete_node(Dllist n);
```

delete and free node n;

```
extern int dll_empty(Dllist l);
```

return 0 if l is empty, 1 otherwise

```
extern Jval dll_val(Dllist);
```

Useful Dllist Macros

`Dllist dll_first(l)`

returns first node in list, `l`, or sentinel if empty

`Dllist dll_last(d)`

returns last node in list, `l`, or sentinel if empty

`Dllist dll_next(n)`

returns next node after `n`, or sentinel if last

`Dllist dll_prev(d)`

returns previous node before `n`, or sentinel if first

`Dllist dll_nil(d)`

returns the sentinel node (`d`)

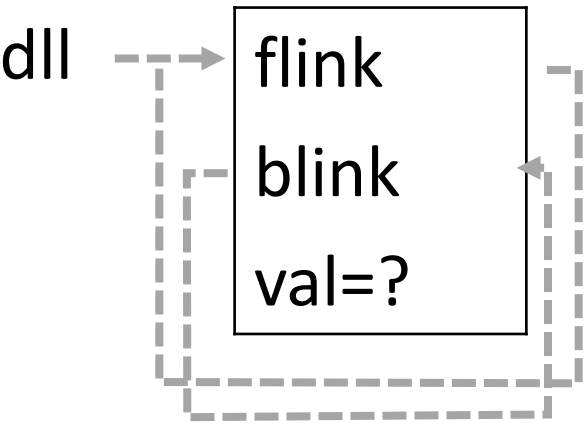
Useful Dllist Macros (cont'd)

```
#define dll_traverse(ptr, list) \  
    for (ptr = list->flink; ptr != list; ptr = ptr->flink)  
    traverse list, 1, with node, ptr, for each iteration
```

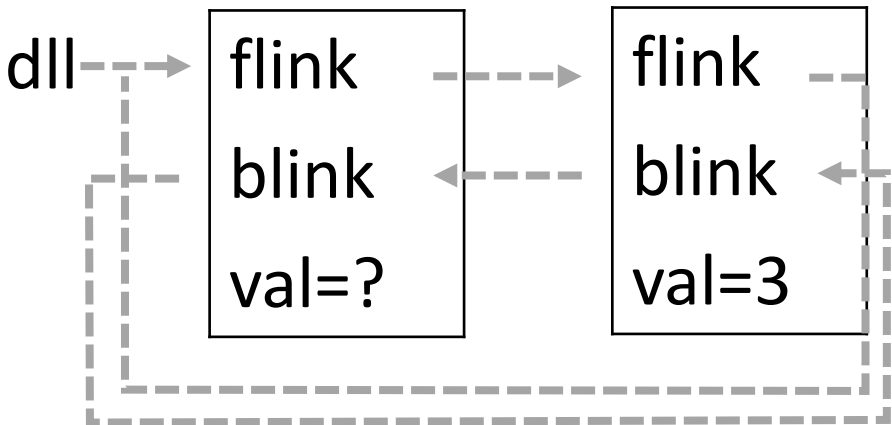
```
#define dll_rtraverse(ptr, list) \  
    for (ptr = list->blink; ptr != list; ptr = ptr->blink)  
    traverse list, 1, in reverse with node, ptr, for each iteration
```



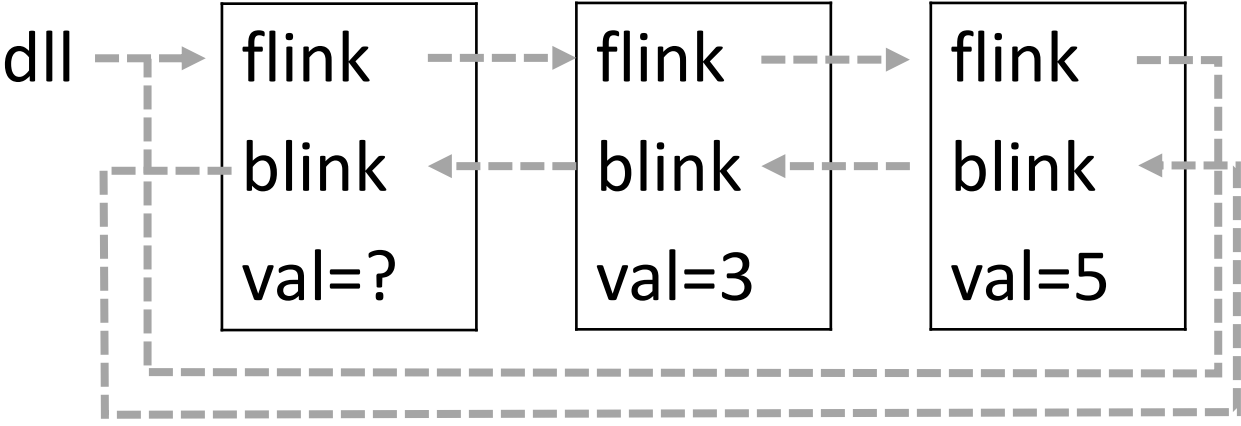
```
dll = new_dllist();
```



```
dll_append(1, new_jval_i(3));
```



```
dll_append(1, new_jval_i(5));
```



```
dll_delete( dll_first(dll) );
```

