Jval tipi

/blugreen/homes/plank/cs360/include/jval.h içerisinde bir **Jval** tipi tanımlanmıştır.Büyük bir union yapısı aşağıdadır:

```
typedef union {
    int i;
    long l;
    float f;
    double d;
    void *v;
    char *s;
    char c;
    unsigned char uc;
    short sh;
    unsigned short ush;
    unsigned int ui;
    int iarray[2];
    float farray[2];
    char carray[8];
    unsigned char ucarray[8];
  } Jval;
```

Genel veri yapılarını kullanacağımız zaman **Jval** den faydalanırız.Önemli olanlar aşağıda mevcuttur:

```
int i;
    float f;
    double d;
    void *v;
    char *s;
    char c;
```

Jval'in güzel özelliklerinden birisi de veri parçasının tipine bağlı olmaksızın,verileri bir parça halinde tutabilmesidir. Dahası jval her zaman 8 bytelik yer kaplar.Jvali union gibi kullanın, bu konuyu union lecture kısmında tartışacağız…

Kurucu Fonksiyonlar

Jval'i oluşturduktan sonra kullanabilirsiniz.Örneğin,

```
Jval j;

  j.i = 4;
```

Serbestçe Jval'e geçebilirsin ve prosedürlerden çağırabilirsin.Bir jval yordam çağrısı dönüş değeri olabilir.

Jval.h , kurucu fonksiyonlar için bir sürü prototip tanımlar.

```
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 *);
```

Yukardakilere belirli türde argüman verildiğinde dönüş tipleri Jval dir.Örneğin: Jval e başlangıçta değeri 4 olan integer atamak istediğimizde şu şekilde yapabiliriz:

```
Jval j;

  j = new_jval_i(4);
```

Şimdi j.i integer 4 oldu.Bunun neden böyle olduğunu daha sonra göreceksin.

Bu kurucu fonksiyonlar, **/blugren/homes/plank/cs360/src/jval.c** ile implement edildi.Örneğin burada **new_jval_i()** yi :

```
Jval new_jval_i(int i) {
  Jval j;
  j.i = i;
  return j;
}
```

Basit bir örnek

Jval i kullanacağımız zaman jval.h ı kütüphanesini ekleriz. Libfdr ders notları içindeki tarif edilmiş olaraklibfdr.a ile sonra bağlarız.

Kodda bir değişiklik olduğunu not et: the typedef:

```
typedef struct {
  char type;
  Jval value;
} Item;
```

Jval yapısı aynı .i, .f , .s alanlarına sahip olduğu zaman, kodun geri kalanı değişmez.

Gördüğün gibi, kod iyi çalışıyor:

```
UNIX> jval_ex
int 4
string Jim
float -33.2
int -2
int 1
Item 0: Type i -- Value: 4
Item 1: Type s -- Value: Jim
Item 2: Type f -- Value: -33.200001
Item 3: Type i -- Value: -2
Item 4: Type i -- Value: 1

Sizeof(Item): 16  UNİX>
```

jval_ex2.c içinde , biz yapıcı fonksiyonları kullanmak için kodu düzenleriz.Asıl değişiklik:

```
  int i2;
  float f;

  ..

  for (i = 0; i < 5; i++) {
    if (get_line(is) != 2) exit(1);

    if (strcmp(is->fields[0], "int") == 0) {
      array[i].type = 'i';
      if (sscanf(is->fields[1], "%d", &i2) != 1) exit(1);
      array[i].value = new_jval_i(i2);

    } else if (strcmp(is->fields[0], "float") == 0) {
      array[i].type = 'f';
      if (sscanf(is->fields[1], "%f", &f) != 1) exit(1);
      array[i].value = new_jval_f(f);

    } else if (strcmp(is->fields[0], "string") == 0) {
      array[i].type = 's';
      array[i].value = new_jval_s(strdup(is->fields[1]));

    } else {
      exit(1);
    }
  }
```

Jval hakkında uyarı kelimeleri

Jval tipinin amacı veri yapılarının genel kullanım şekli tıpkı dllist, red-black de olduğu gibidir.Jval başka bir neden için Jval i kodunda kullanmazsın.Size ne zaman kullanıldığını söyleyeceğim.
Özellikle, Jval int in yerine kullanılabilir çünkü çalışır.Ama kodun okunabillirliği azalır.Aşağıda standart giriş üzerinde girilen integer sayıların ortalamasını alan kötü bir kod örneği mevcuttur(badavg.c):

```
main()
{
  Jval total;
  Jval j;
  Jval n;

  n.i = 0;
  total.i = 0;

  while (scanf("%d", &(j.i)) == 1) {
    total.i += j.i;
    n.i++;
  }

  total.d = ((double) total.i) / ((double) n.i);
```

```
  printf("Average = %lf\n", total.d);
}
```

Yukarıdaki kod çalışır ve bir double ve bir in tın 2 sini de total de kullanmak akıllıca bir yöntemdir. Fakat Jval i her kullandığımızda kötü olur ve sen bunları bu yöntemle kullanırsan , cezalandırılacaksın.

(Dikkat etmen gerek bir bölüm de , kodun goodavg.c deki gibi görünmesi gerekliliğidir.)

```
main()
{
  int total;
  int j;
  int n;

  n = 0;
  total = 0;

  while (scanf("%d", &j) == 1) {
    total += j;
    n++;
  }

  if (n == 0) exit(1);

  printf("Average = %lf\n", ((double) total)/((double) n));
}
```

(Bu kullanım daha iyi görünür,)

Erişim fonksiyonları

Erişim fonksiyonları jval.h/jval.c içine yerleştirilir. Basitçe erişim fonksiyonları erişim alanından sonra bir fonksiyon çağırarak Jval in istenen değerini almana izin verir.Niçin bunu yapmak istiyoruz? Kurucu fonksiyonları gibi , bazı koşullarla kullanım kolaylığı sağlar.Erişim fonksiyonları aşağıdaki gibidir:

```
extern int    jval_i(Jval);
extern long   jval_l(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);
...
```

Örneğin, j.i kullanımıyla jval_i(j) fonksiyonunu çağırmak aynıdır.

JNULL

Son olarak , jval.h global değişken JNULL u içerir. Char* veya void* için NULL kullandığımız zaman, bunu yaparız.