Appendix

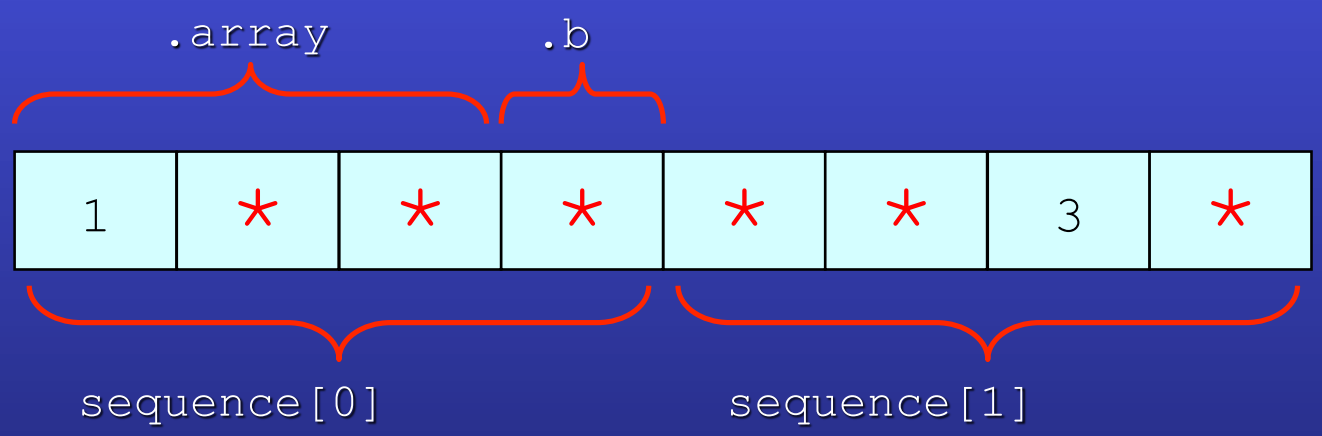
arrays and structs

- **arrays and struct may contain each other**
  - **[int] and .identifier designators can be combined**

struct containing array

```
struct s
{
    int array[3];
    int b;
};
```

```
struct s sequence[] =
    {
        [0].array = { 1 },
        [1].array[2] = 3
    };
```



* default value

rectangular arrays

- only the top-level array decays into a pointer
  - the size of sub arrays remains part of the type

```
void print(int nrows, int matrix[2][3]);
void print(int nrows, int matrix[ ][3]);
void print(int nrows, int (*matrix)[3]);
```

equivalent

```
int main(void)
{
    int grid[2][3] = {{0,1,2},{3,4,5}};
    ...
    print(2, grid);
}
```

```
void illegal(int matrix[ ][ ]) ...
```

argument picture

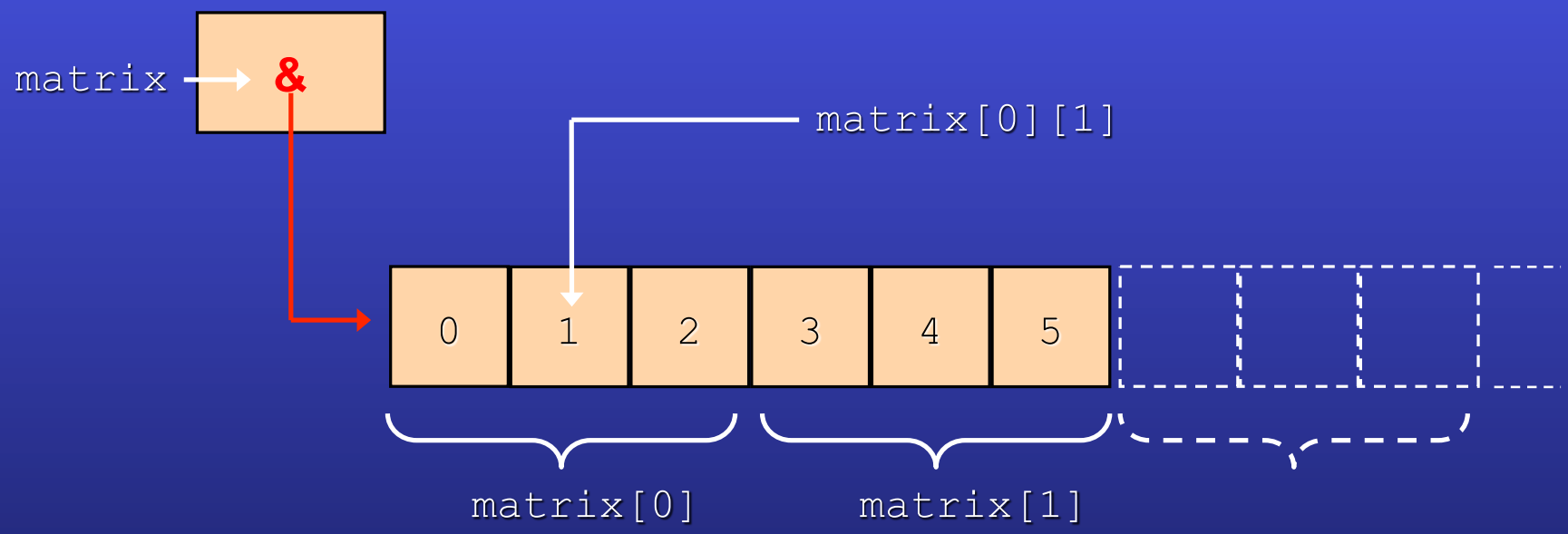
```
int (*matrix)[3]
```

```
int (*matrix)[3]

int (*matrix)[3]
```

matrix is a pointer
to zero, one, or more
array of three ints

matrix points to a *single* chunk of memory



ragged arrays

- an array of pointers can mimic a 2d array
  - each pointer points to an array
  - aka Illiffe vector aka dope vector

note this is `int*ragged[2]` and not `int(*ragged)[2]`

```
void print(int nrows, int ncols, int * ragged[2]);
void print(int nrows, int ncols, int * ragged[ ]);
void print(int nrows, int ncols, int * * ragged);
```

equivalent

```
int main(void)
{
    int vec1[] = { 0, 1, 2 };
    int vec2[] = { 3, 4, 5 };
    int * grid[2] = { vec1, vec2 };

    print(2, 3, grid);
}
```

argument picture

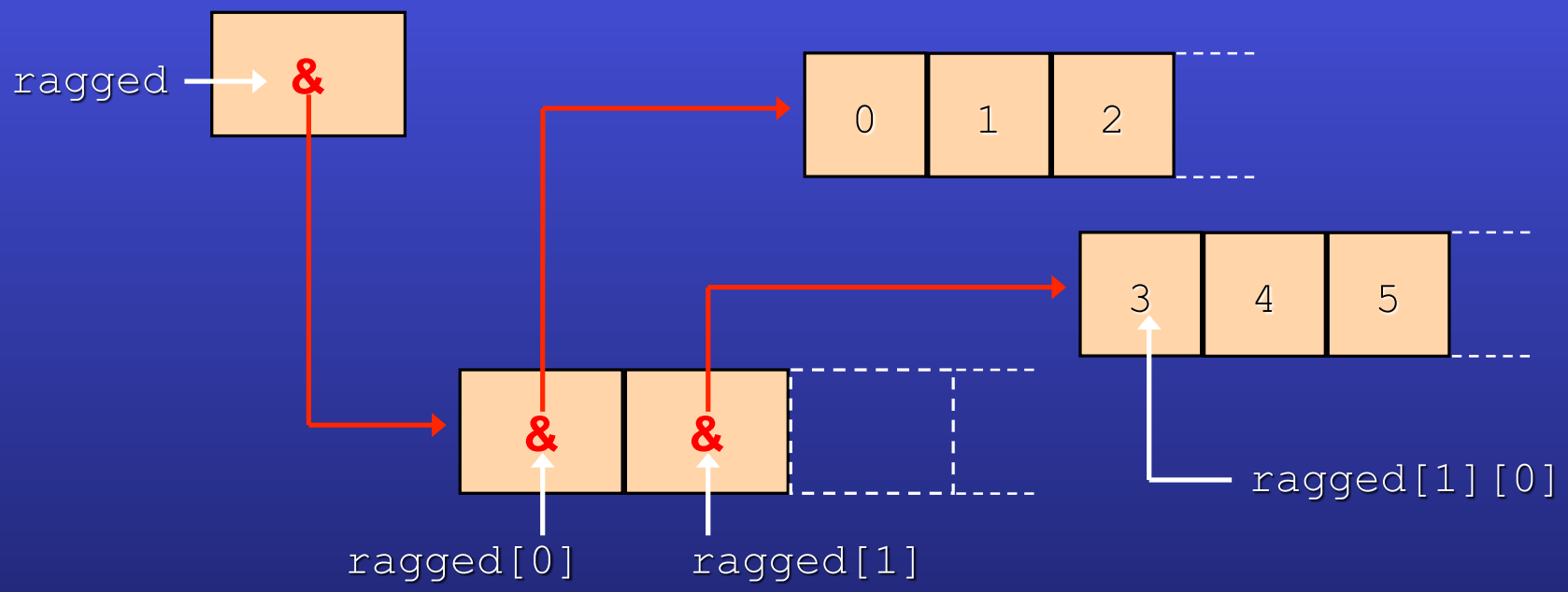
```
int * ragged[]
```

```
int * ragged[]

int * ragged[]

int * ragged[]
```

ragged is an array
of
pointers
to zero, one, or more
int



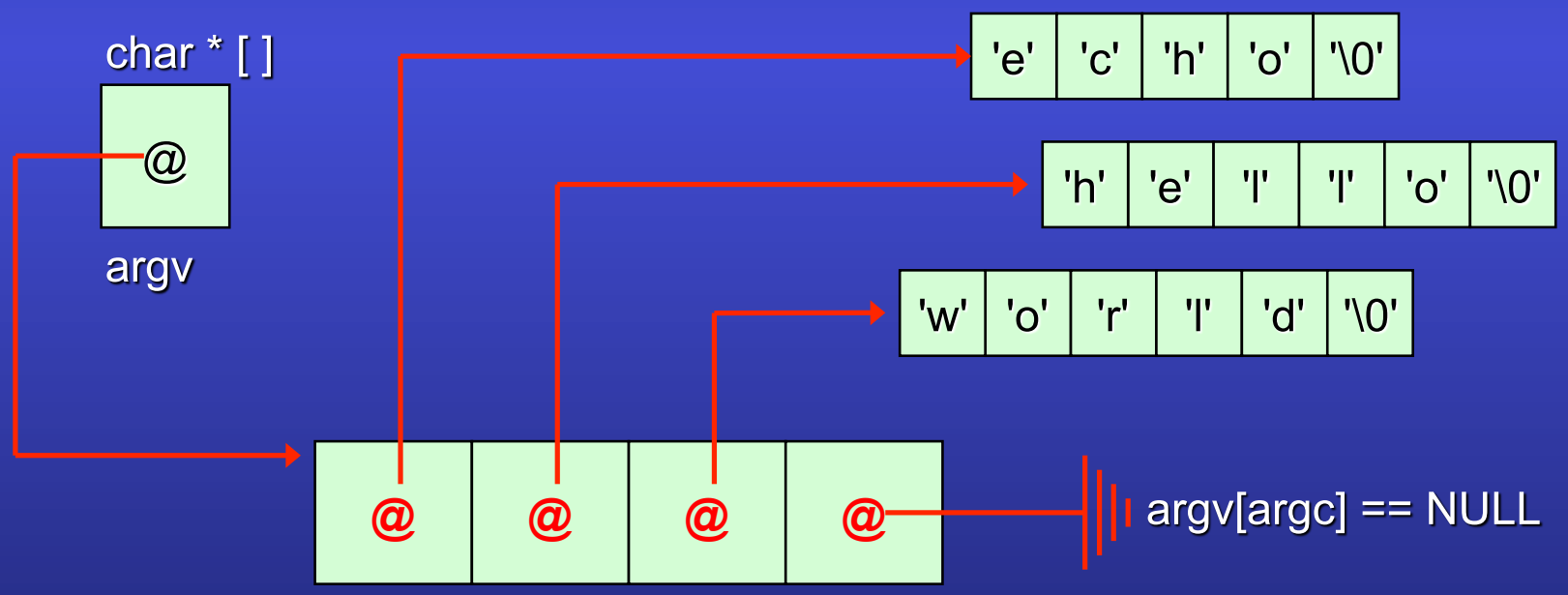
ragged array example

echo.c

```
#include <stdio.h>

int main(int argc, char * argv[])
{
    for (int at = 0; at != argc; at++)
        printf("%s ", argv[at]);
    putchar('\n');
}
```

```
>echo hello world
```



variable length arrays

- make multi-dimensional array parameters much more useful and reusable
  - several restrictions (see notes)

```
void print(int n, int m,  int matrix[ n ] [ m ] );
```

```
void print(int n, int m,  int matrix[ n ] [ m ] )
{
    ...
}
```

n and m must be declared before matrix

```
int main(void)
{
    int matrix[][3] = {{ 0, 1, 2 }, { 3, 4, 5 }};
    print(2, 3, matrix);
}
```

#line

- **#line changes the apparent line number and name of the source file**
  - **useful only to code generators**

s-char == any character except " \ or newline

```
# line digit-sequence "s-chars"opt
```

the line number (required)

the filename (optional)

example.c

```
#line 999 "fake.c"

bug++;
```

```
>gcc example.c
fake.c:1000: error: syntax error before '++' token
```