C/C++ Refresher

ESE532

Makefile

target: dependencies
    Action

$@: the filename representing the target
$^: the filenames of all the prerequisites

```
CXX = g++
CXXFLAGS = -std=c++11 -Wall -O3 -mcpu=native -fno-tree-vectorize -DNDEBUG -pthread
LDFLAGS = -lpthread -pthread
INCLUDES = -I./common

EXECUTABLES = baseline neon_filter example
SRC = $(wildcard src/*.cpp common/*.cpp)
EXAMPLE_SRC = $(wildcard neon_example/*.cpp common/*.cpp)

.PHONY: clean

all: $(EXECUTABLES)
    @echo $(EXECUTABLES) compiled!

baseline: $(SRC)
    $(CXX) $(CXXFLAGS) $(INCLUDES) $^ -o $@ $(LDFLAGS)

neon_filter: $(SRC)
    $(CXX) -DVECTORIZED $(CXXFLAGS) $(INCLUDES) $^ -o $@ $(LDFLAGS)

example: $(EXAMPLE_SRC)
    $(CXX) $(CXXFLAGS) $(INCLUDES) $^ -o $@ $(LDFLAGS)
```

<Example of Makefile>

Makefile

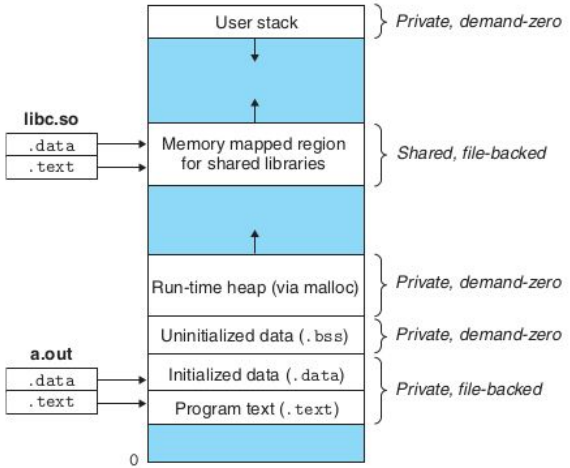
VPP := v++
TARGET := hw

```
${HOST}: ${HOST_SRC}
    g++ ${GCC_OPTS} -I ${SOURCE_DIR} -o $@ $+
    @echo 'Compiled Host Executable: ${HOST_EXE}'

${XOS}: ${KERNEL_SRC}
    @${RM} $@
    ${VPP} -t ${TARGET} --config ${CONFIG_DIR}/design.cfg \
        --log_dir ${LOG_DIR} \
        --report_dir ${REPORT_DIR} \
        --platform ${AWS_PLATFORM} \
        --compile --kernel ${NAME} \
        -D ${N} \
        -I ${SOURCE_DIR} -o $@ $+
    mv ${BUILD_DIR}/*compile_summary ${REPORT_DIR}/${NAME}.${TARGET}/

${XCLBIN}: ${XOS}
    ${VPP} -t ${TARGET} --config ${CONFIG_DIR}/design.cfg \
        --log_dir ${LOG_DIR} \
        --report_dir ${REPORT_DIR} \
        --platform ${AWS_PLATFORM} \
        --link -o $@ $+
    mv ${BUILD_DIR}/*link_summary ${REPORT_DIR}/${NAME}.${TARGET}/

${EMCONFIG_FILE}:
    ${EMCONFIGUTIL} --platform ${AWS_PLATFORM} --od ${BUILD_DIR}

emulate: ${HOST} ${XCLBIN} ${EMCONFIG_FILE}
    echo Running host code with kernel...
    XCL_EMULATION_MODE=${TARGET} ./${HOST} ${XCLBIN}
    echo Finished run
```

<Example of Makefile used in Vitis flow>

Stack and Heap memory

- stack: stores local variables
- heap: dynamic memory for programmer to allocate
- data: stores global variables, separated into initialized and uninitialized
- text: stores the code being executed

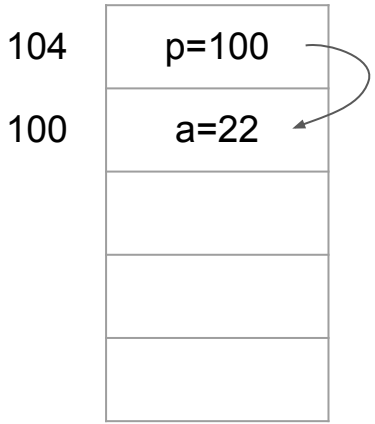


<User address space>*

*Figure from “Computer Systems, A Programmer’s Perspective”

Pointers

What are the print outputs?



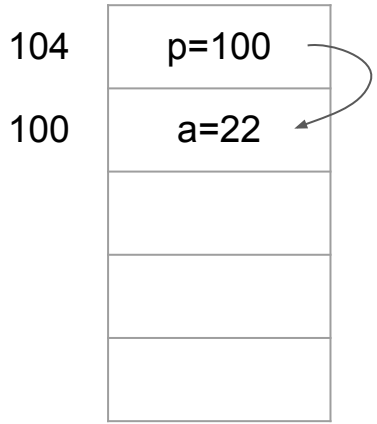

```c
int main(){
    int a;
    int *p;
    a = 22;
    p = &a;
    printf("p = %x\n", p);
    printf("&a = %x\n", &a);
    printf("&p = %x\n", &p);
    printf("*p = %x\n", *p);
    printf("p+1 = %x\n", p+1);
```

<Pointers basics, code and memory map>

Pointers

Answer:

```
p = 100
&a = 100
&p = 104
*p = 16 // hex
p+1 = 104
```




```c
int main(){
    int a;
    int *p;
    a = 22;
    p = &a;
    printf("p = %x\n", p);
    printf("&a = %x\n", &a);
    printf("&p = %x\n", &p);
    printf("*p = %x\n", *p);
    printf("p+1 = %x\n", p+1);
```

<Pointers basics, code and memory map>

Pointers

What are the print outputs?

```
3  int main(){
4      int a = 1025;
5      int *p_0;
6      p_0 = &a;
7
8      char *p_1;
9      p_1 = (char*)p_0; // typecasting
10
11     printf("size of int = %d\n", sizeof(int)); // 4
12     printf("size of char = %d\n", sizeof(char)); // 1
13     printf("address = %d, value = %d\n", p_1, *p_1);
14     printf("address = %d, value = %d\n", p_1+1, *(p_1+1));
15
16     // hint: 1025 = 00000000 00000000 00000100 00000001
```

<Pointers typecasting, code>

Pointers

Answer:

```
address = 100, value = 1
address = 101, value = 4
```

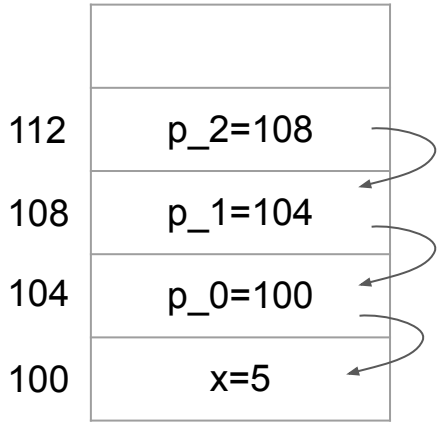
```c
int main(){
    int a = 1025;
    int *p_0;
    p_0 = &a;

    char *p_1;
    p_1 = (char*)p_0; // typecasting

    printf("size of int = %d\n", sizeof(int)); // 4
    printf("size of char = %d\n", sizeof(char)); // 1
    printf("address = %d, value = %d\n", p_1, *p_1);
    printf("address = %d, value = %d\n", p_1+1, *(p_1+1));

    // hint: 1025 = 00000000 00000000 00000100 00000001
```

<Pointers typecasting, code>

Pointers

What are the print outputs?

Which line outputs 5?



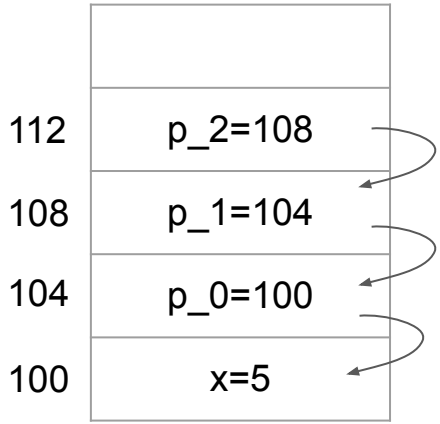

```c
int main(){
    int x = 5;
    int *p_0;
    int **p_1;
    int ***p_2;

    p_0 = &x;
    p_1 = &p_0;
    p_2 = &p_1;

    printf("&x = %d\n", &x);
    printf("&p_0 = %d\n", &p_0);
    printf("&p_1 = %d\n", &p_1);
    printf("&p_2 = %d\n", &p_2);

    printf("*p_0 = %d\n", *p_0);
    printf("*p_1 = %d\n", *p_1);
    printf("**p_1 = %d\n", **p_1);
    printf("*p_2 = %d\n", *p_2);
    printf("**p_2 = %d\n", **p_2);
    printf("***p_2 = %d\n", ***p_2);
```

<Pointers to pointers, code and memory map>

Pointers

Answer:

```
&x  = 100
&p_0  = 104
&p_1  = 108
&p_2  = 112

*p_0  = 5
*p_1  = 100
**p_1  = 5
*p_2  = 104
**p_2  = 100
***p_2  = 5
```




```c
int main(){
    int x = 5;
    int *p_0;
    int **p_1;
    int ***p_2;

    p_0 = &x;
    p_1 = &p_0;
    p_2 = &p_1;

    printf("&x = %d\n", &x);
    printf("&p_0 = %d\n", &p_0);
    printf("&p_1 = %d\n", &p_1);
    printf("&p_2 = %d\n", &p_2);

    printf("*p_0 = %d\n", *p_0);
    printf("*p_1 = %d\n", *p_1);
    printf("**p_1 = %d\n", **p_1);
    printf("*p_2 = %d\n", *p_2);
    printf("**p_2 = %d\n", **p_2);
    printf("***p_2 = %d\n", ***p_2);
```

<Pointers to pointers, code and memory map>

Pointers

What are the print outputs?

116	A[4] = 5
112	A[3] = 4
108	A[2] = 3
104	A[1] = 2
100	A[0] = 1

```
3  int main(){
4      int A[] = {1,2,3,4,5};
5
6      printf("A = %d\n", A);
7      printf("&A = %d\n", &A);
8      printf("&A[0] = %d\n", &A[0]);
9      printf("A[0] = %d\n", A[0]);
10     printf("*A = %d\n", *A);
11     printf("A+1 = %d\n", A+1);
12     printf("A[1] = %d\n", A[1]);
13     printf("*(A+1) = %d\n", *(A+1));
```

<Pointers and arrays, code and memory map>

Pointers

Answer:

```
A = 100
&A = 100
&A[0] = 100
A[0] = 1
*A = 1
A+1 = 104
A[1] = 2
*(A+1) = 2
```

116	A[4] = 5
112	A[3] = 4
108	A[2] = 3
104	A[1] = 2
100	A[0] = 1

```
3  int main(){
4      int A[] = {1,2,3,4,5};
5
6      printf("A = %d\n", A);
7      printf("&A = %d\n", &A);
8      printf("&A[0] = %d\n", &A[0]);
9      printf("A[0] = %d\n", A[0]);
10     printf("*A = %d\n", *A);
11     printf("A+1 = %d\n", A+1);
12     printf("A[1] = %d\n", A[1]);
13     printf("*(A+1) = %d\n", *(A+1));
```

<Pointers and arrays, code and memory map>

Pointers

What are the print outputs?

120	A[1][2] = 6
116	A[1][1] = 5
112	A[1][0] = 4
108	A[0][2] = 3
104	A[0][1] = 2
100	A[0][0] = 1

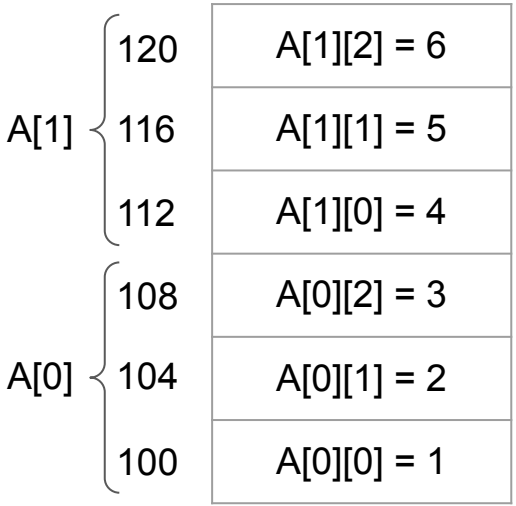
```
3  int main(){
4      int A[2][3] = {{1,2,3},
5                     {4,5,6}};
6
7      printf("A = %d\n", A);
8      printf("&A = %d\n", &A);
9      printf("&A[0] = %d\n", &A[0]);
10
11     printf("A[0] = %d\n", A[0]);
12     printf("&A[0][0] = %d\n", &A[0][0]);
13     printf("*A = %d\n", *A);
14
15     printf("A[0][0] = %d\n", A[0][0]);
16
17     printf("A+1 = %d\n", A+1);
18     printf("A[1] = %d\n", A[1]);
19     printf("*(A+1) = %d\n", *(A+1));
20
21     printf("*(A+1)+2 = %d\n", *(A+1)+2);
22     printf("A[1]+2 = %d\n", A[1]+2);
23     printf("&A[1][2] = %d\n", &A[1][2]);
24
25     printf("*(*(A+1)+2) = %d\n", *(*(A+1)+2));
26     printf("*(A[1]+2) = %d\n", *(A[1]+2));
27     printf("A[1][2] = %d\n", A[1][2]);
```

<Pointers and 2D arrays, code and memory map>

Pointers

Answer:

```
A = 100
&A = 100
&A[0] = 100

A[0] = 100
&A[0][0] = 100
*A = 100

A[0][0] = 1

A+1 = 112
A[1] = 112
*(A+1) = 112

*(A+1)+2 = 120
A[1]+2 = 120
&A[1][2] = 120

*A+1 = 6
*(A[1]+2) = 6
A[1][2] = 6
```

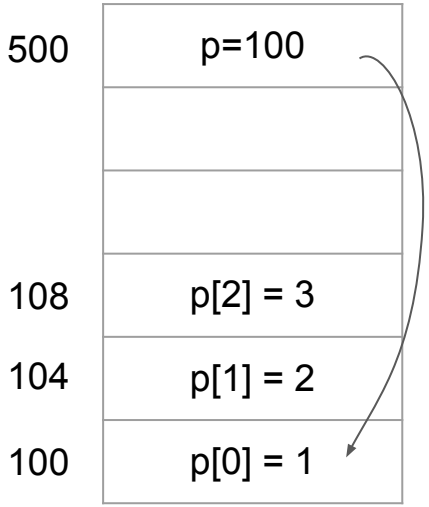



```c
int main(){
    int A[2][3] = {{1,2,3},
                   {4,5,6}};

    printf("A = %d\n", A);
    printf("&A = %d\n", &A);
    printf("&A[0] = %d\n", &A[0]);

    printf("A[0] = %d\n", A[0]);
    printf("&A[0][0] = %d\n", &A[0][0]);
    printf("*A = %d\n", *A);

    printf("A[0][0] = %d\n", A[0][0]);

    printf("A+1 = %d\n", A+1);
    printf("A[1] = %d\n", A[1]);
    printf("*(A+1) = %d\n", *(A+1));

    printf("*(A+1)+2 = %d\n", *(A+1)+2);
    printf("A[1]+2 = %d\n", A[1]+2);
    printf("&A[1][2] = %d\n", &A[1][2]);

    printf("*(*(A+1)+2) = %d\n", *(*(A+1)+2));
    printf("*(A[1]+2) = %d\n", *(A[1]+2));
    printf("A[1][2] = %d\n", A[1][2]);
```

<Pointers and 2D arrays, code and memory map>

Pointers

What are the print outputs?



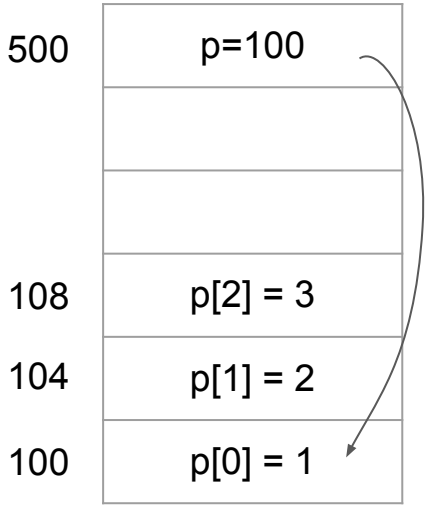

```c
int main(){
    int a;
    int *p;
    p = (int*)malloc(3*sizeof(int));
    p[0] = 1;
    p[1] = 2;
    p[2] = 3;

    printf("&p = %p\n", &p);
    printf("p = %p\n", p);

    printf("&p[0] = %p\n", &p[0]);
    printf("&p[1] = %p\n", &p[1]);
    printf("&p[2] = %p\n", &p[2]);

    printf("p[0] = %d\n", p[0]);
    printf("p[1] = %d\n", p[1]);
    printf("p[2] = %d\n", p[2]);

    free(p);
```

<Pointers and dynamic memory, code and memory map>

Pointers

Answer:

```
&p = 500 // stack
p = 100 // heap

&p[0] = 100 // heap
&p[1] = 104 // heap
&p[2] = 108 // heap

p[0] = 1
p[1] = 2
p[2] = 3
```




```c
int main(){
    int a;
    int *p;
    p = (int*)malloc(3*sizeof(int));
    p[0] = 1;
    p[1] = 2;
    p[2] = 3;

    printf("&p = %p\n", &p);
    printf("p = %p\n", p);

    printf("&p[0] = %p\n", &p[0]);
    printf("&p[1] = %p\n", &p[1]);
    printf("&p[2] = %p\n", &p[2]);

    printf("p[0] = %d\n", p[0]);
    printf("p[1] = %d\n", p[1]);
    printf("p[2] = %d\n", p[2]);

    free(p);
```

<Pointers and dynamic memory, code and memory map>

Common memory related mistakes in C

What is wrong?

```
1   /* Create an nxm array */
2   int **makeArray1(int n, int m)
3   {
4       int i;
5       int **A = (int **)Malloc(n * sizeof(int));
6
7       for (i = 0; i < n; i++)
8           A[i] = (int *)Malloc(m * sizeof(int));
9       return A;
10  }
```

<Assuming That Pointers and the Objects They Point to Are the Same Size>*

*Code snippet from “Computer Systems, A Programmer’s Perspective”

Common memory related mistakes in C

What is wrong?

```
1  int *search(int *p, int val)
2  {
3      while (*p && *p != val)
4          p += sizeof(int); /* Should be p++ */
5      return p;
6  }
```

<Misunderstanding Pointer Arithmetic>*

*Code snippet from “Computer Systems, A Programmer’s Perspective”

References

- mycodeschool, Pointers in C/C++
- “Computer Systems, A Programmer’s Perspective”