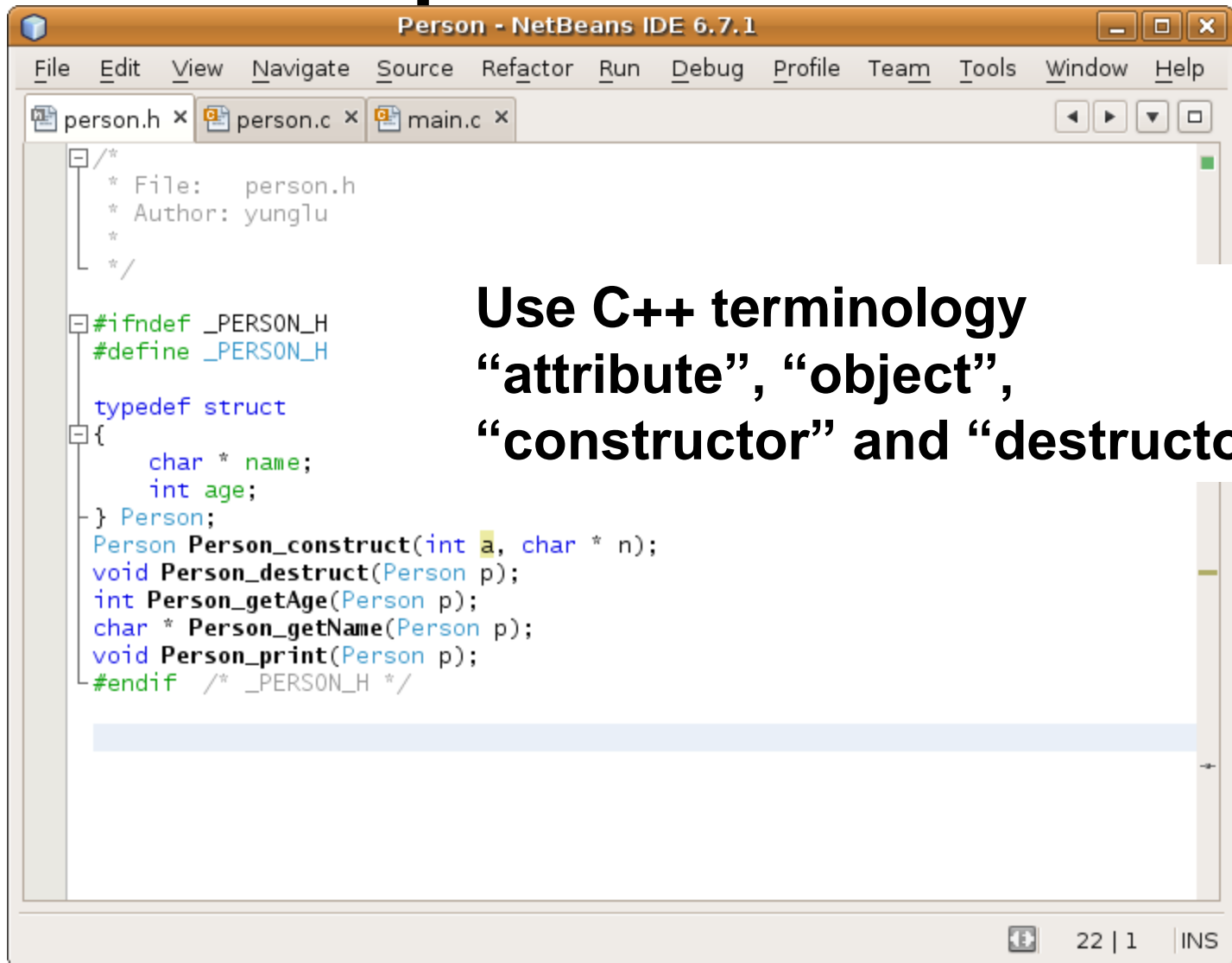


# **Structure with Pointer and Memory Allocation**

**Yung-Hsiang Lu**

# person.h



```
/*
 * File:   person.h
 * Author: yunglu
 *
 */

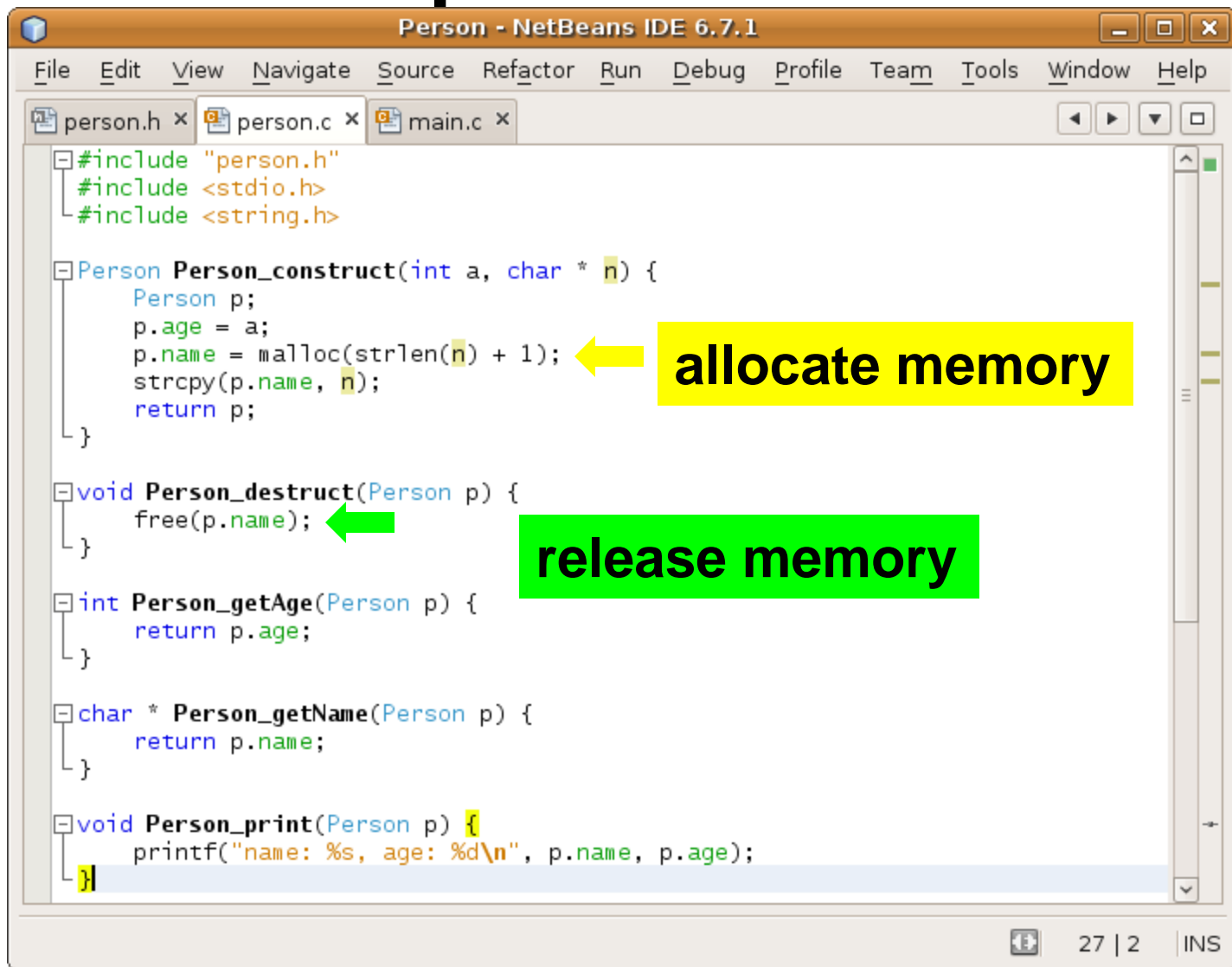
#ifndef _PERSON_H
#define _PERSON_H

typedef struct
{
    char * name;
    int age;
} Person;

Person Person_construct(int a, char * n);
void Person_destruct(Person p);
int Person_getAge(Person p);
char * Person_getName(Person p);
void Person_print(Person p);
#endif /* _PERSON_H */
```

**Use C++ terminology  
“attribute”, “object”,  
“constructor” and “destructor”**

# person.c



The image shows a NetBeans IDE 6.7.1 window with the file 'person.c' open. The code defines a 'Person' struct and several functions. A yellow arrow points to the 'malloc' call in 'Person\_construct', and a green arrow points to the 'free' call in 'Person\_destruct'. A yellow box labeled 'allocate memory' is next to the first arrow, and a green box labeled 'release memory' is next to the second arrow.

```
#include "person.h"
#include <stdio.h>
#include <string.h>

Person Person_construct(int a, char * n) {
    Person p;
    p.age = a;
    p.name = malloc(strlen(n) + 1);
    strcpy(p.name, n);
    return p;
}

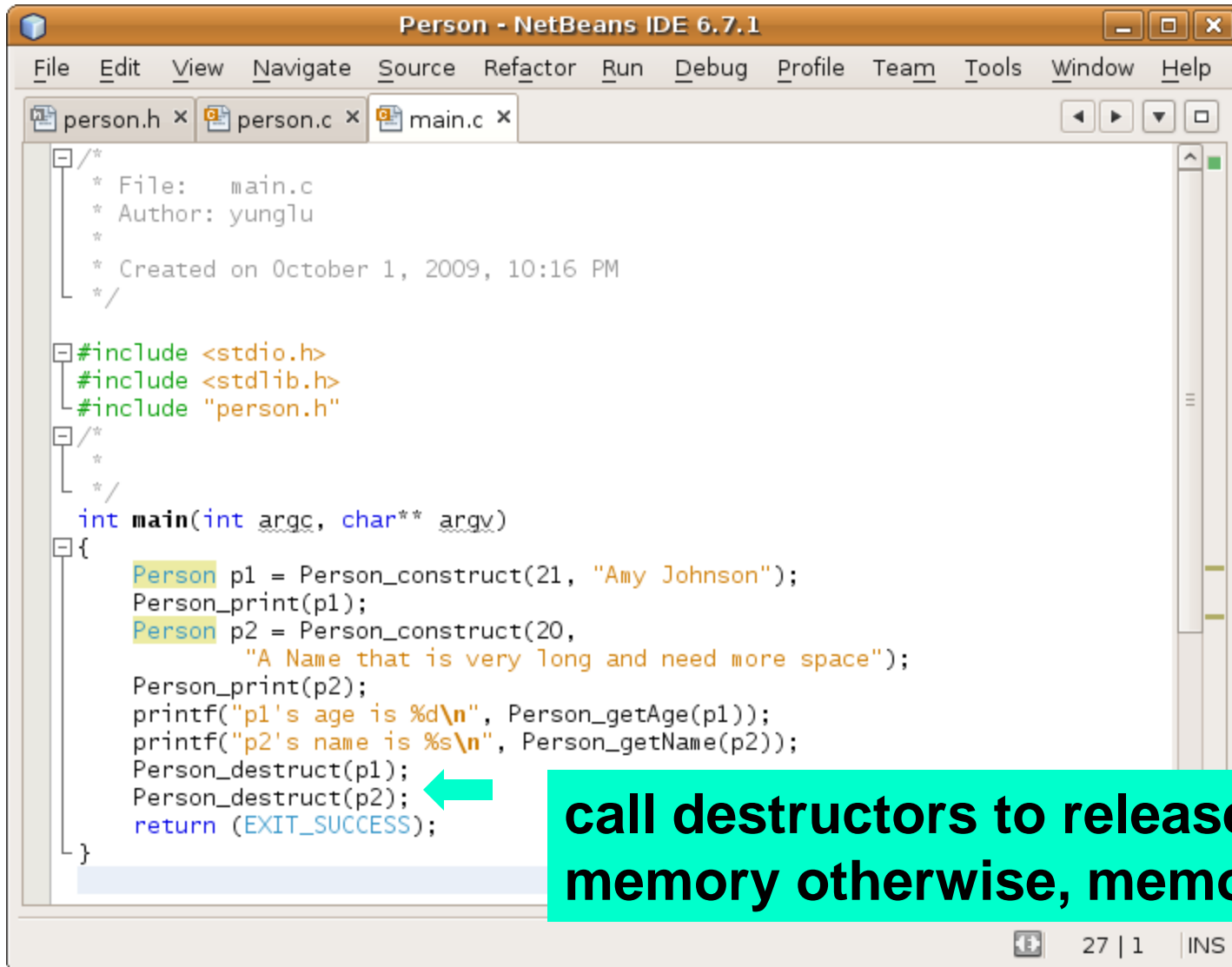
void Person_destruct(Person p) {
    free(p.name);
}

int Person_getAge(Person p) {
    return p.age;
}

char * Person_getName(Person p) {
    return p.name;
}

void Person_print(Person p) {
    printf("name: %s, age: %d\n", p.name, p.age);
}
```

# main.c



```
Person - NetBeans IDE 6.7.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
person.h x person.c x main.c x
/*
 * File:   main.c
 * Author: yunglu
 *
 * Created on October 1, 2009, 10:16 PM
 */

#include <stdio.h>
#include <stdlib.h>
#include "person.h"

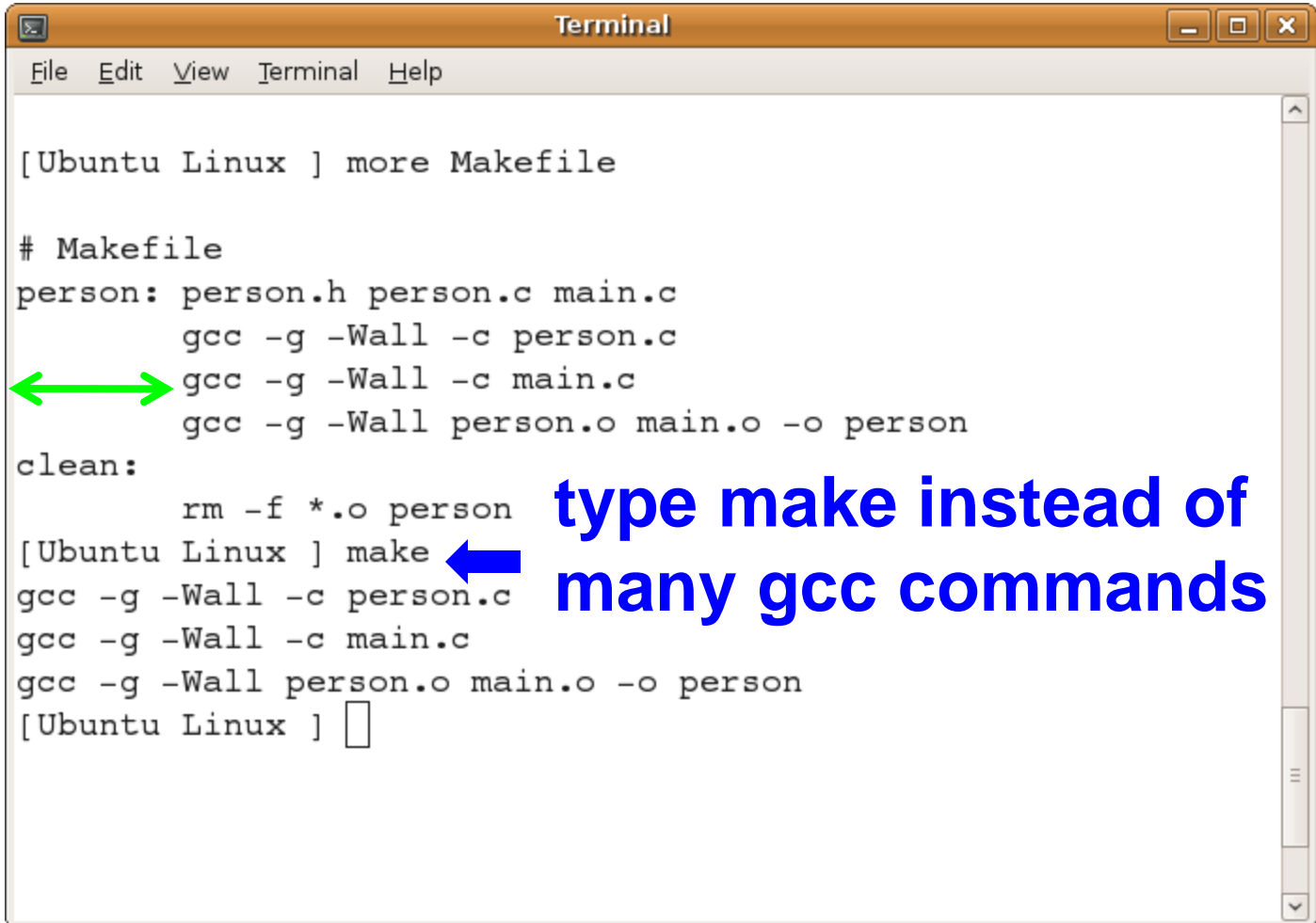
/*
 *
 */
int main(int argc, char** argv)
{
    Person p1 = Person_construct(21, "Amy Johnson");
    Person_print(p1);
    Person p2 = Person_construct(20,
        "A Name that is very long and need more space");
    Person_print(p2);
    printf("p1's age is %d\n", Person_getAge(p1));
    printf("p2's name is %s\n", Person_getName(p2));
    Person_destruct(p1);
    Person_destruct(p2);
    return (EXIT_SUCCESS);
}
```

**call destructors to release memory otherwise, memory leak**

27 | 1 | INS

# Makefile

**tab, not  
space**



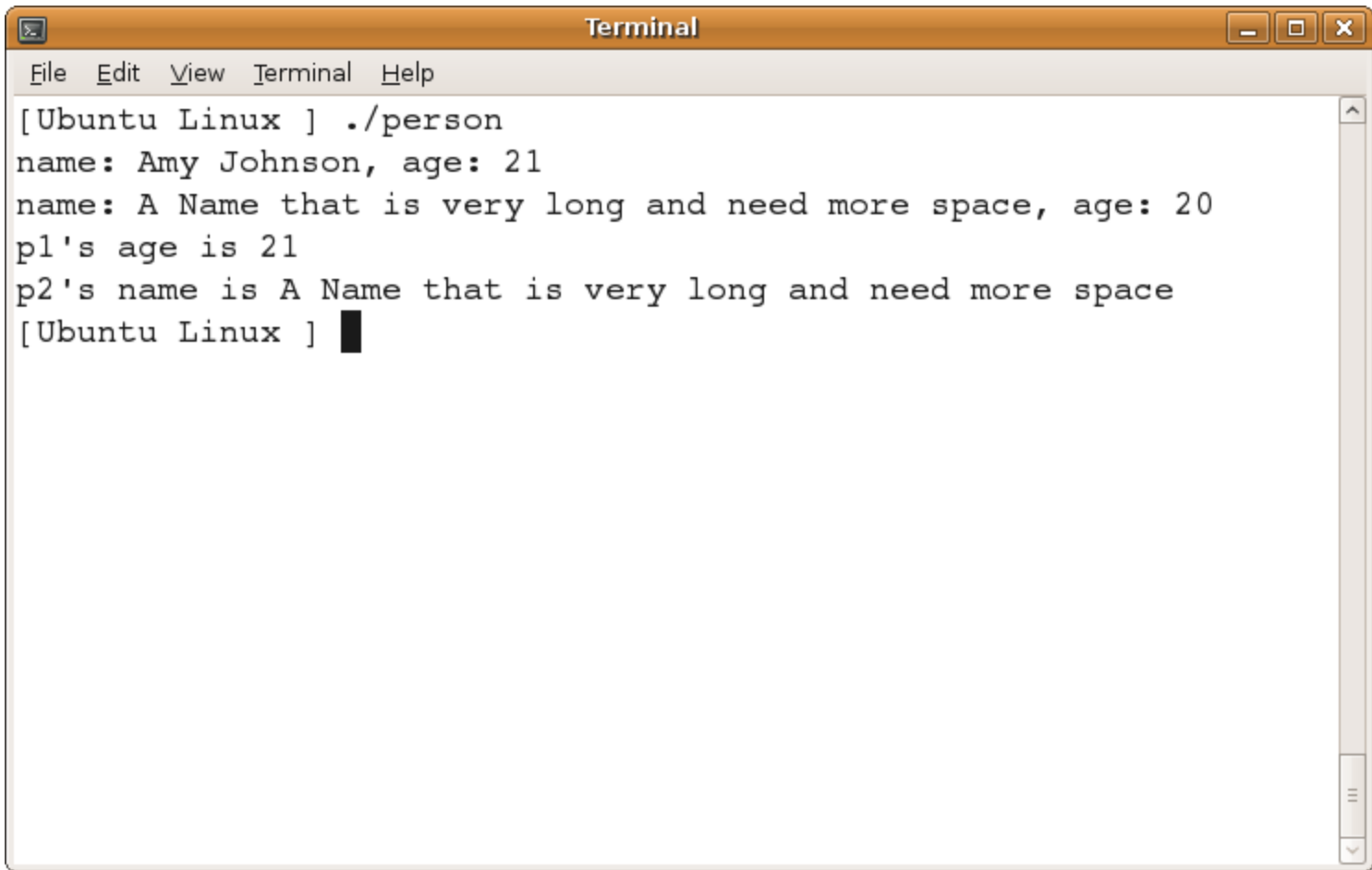
A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Help). The terminal shows the command `more Makefile` and the contents of the Makefile. The Makefile defines a target `person` with dependencies `person.h`, `person.c`, and `main.c`. The rules for `person` are: `gcc -g -Wall -c person.c`, `gcc -g -Wall -c main.c`, and `gcc -g -Wall person.o main.o -o person`. The `clean` target has the rule `rm -f *.o person`. Below the Makefile, the terminal shows the command `make` being executed, which runs the same gcc commands as defined in the Makefile. A green double-headed arrow points from the text "tab, not space" to the tab character in the gcc commands. A blue arrow points from the text "type make instead of many gcc commands" to the `make` command.

```
[Ubuntu Linux ] more Makefile

# Makefile
person: person.h person.c main.c
    gcc -g -Wall -c person.c
    gcc -g -Wall -c main.c
    gcc -g -Wall person.o main.o -o person
clean:
    rm -f *.o person
[Ubuntu Linux ] make
gcc -g -Wall -c person.c
gcc -g -Wall -c main.c
gcc -g -Wall person.o main.o -o person
[Ubuntu Linux ]
```

**type make instead of  
many gcc commands**

# Output

A screenshot of a Linux terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", and "Help". The terminal content shows the execution of a program with the following output:

```
[Ubuntu Linux ] ./person
name: Amy Johnson, age: 21
name: A Name that is very long and need more space, age: 20
p1's age is 21
p2's name is A Name that is very long and need more space
[Ubuntu Linux ]
```

The cursor is at the end of the last line. The window includes standard Linux window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.