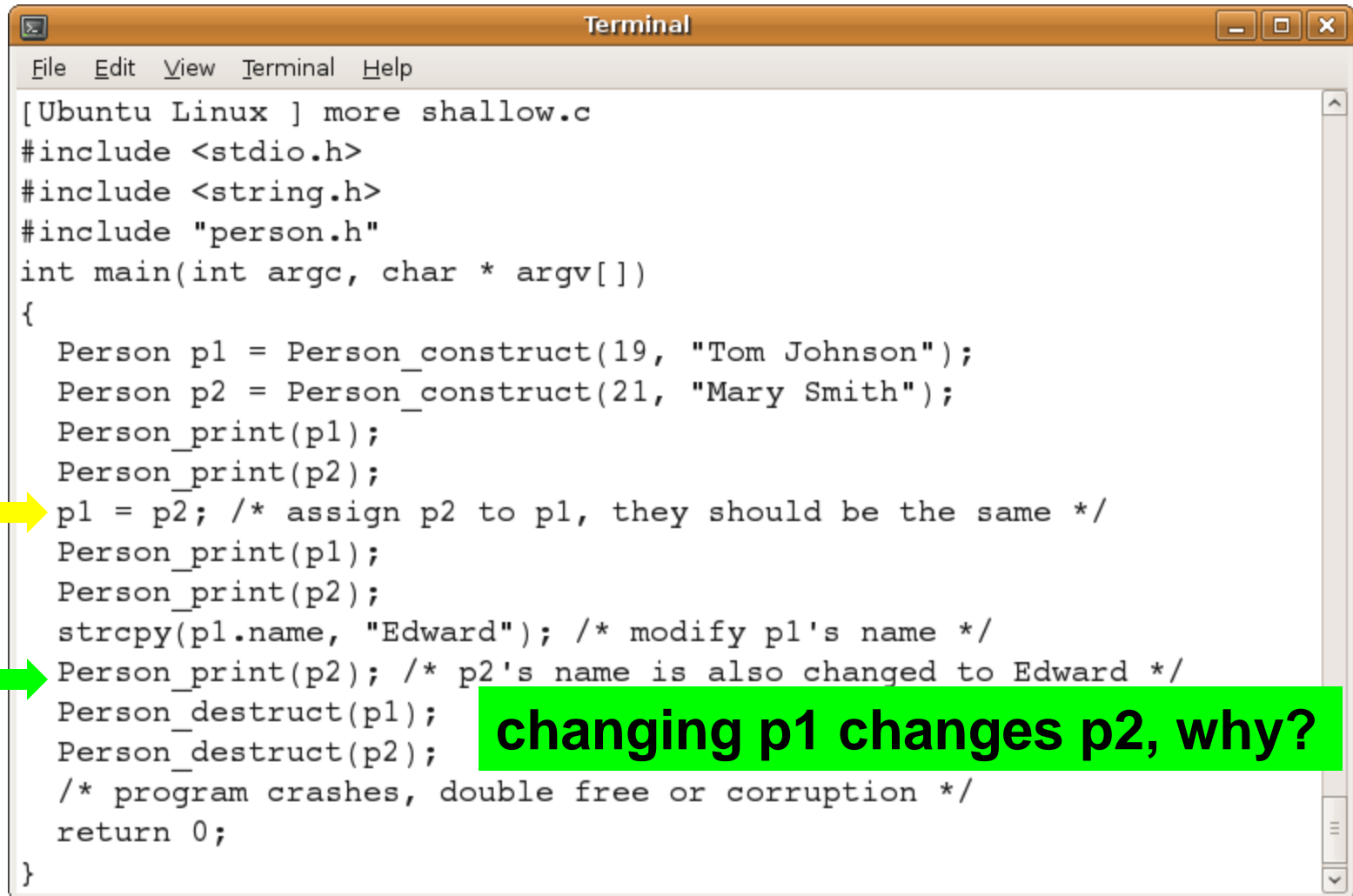


# Deep Copy

Yung-Hsiang Lu

# Shallow Copy and the Problems

# p1 = p2



```
Terminal
File Edit View Terminal Help
[Ubuntu Linux ] more shallow.c
#include <stdio.h>
#include <string.h>
#include "person.h"
int main(int argc, char * argv[])
{
    Person p1 = Person_construct(19, "Tom Johnson");
    Person p2 = Person_construct(21, "Mary Smith");
    Person_print(p1);
    Person_print(p2);
    p1 = p2; /* assign p2 to p1, they should be the same */
    Person_print(p1);
    Person_print(p2);
    strcpy(p1.name, "Edward"); /* modify p1's name */
    Person_print(p2); /* p2's name is also changed to Edward */
    Person_destruct(p1);
    Person_destruct(p2);
    /* program crashes, double free or corruption */
    return 0;
}
```

# Variable Assignments

```
int x = 5;
```

```
int y = 10;
```

```
x = y;    /* x is 10 now */
```

```
x = 20;   /* y is still 10 */
```

```
y = 0;    /* x is still 20 */
```

```
Person p1 = ...
```

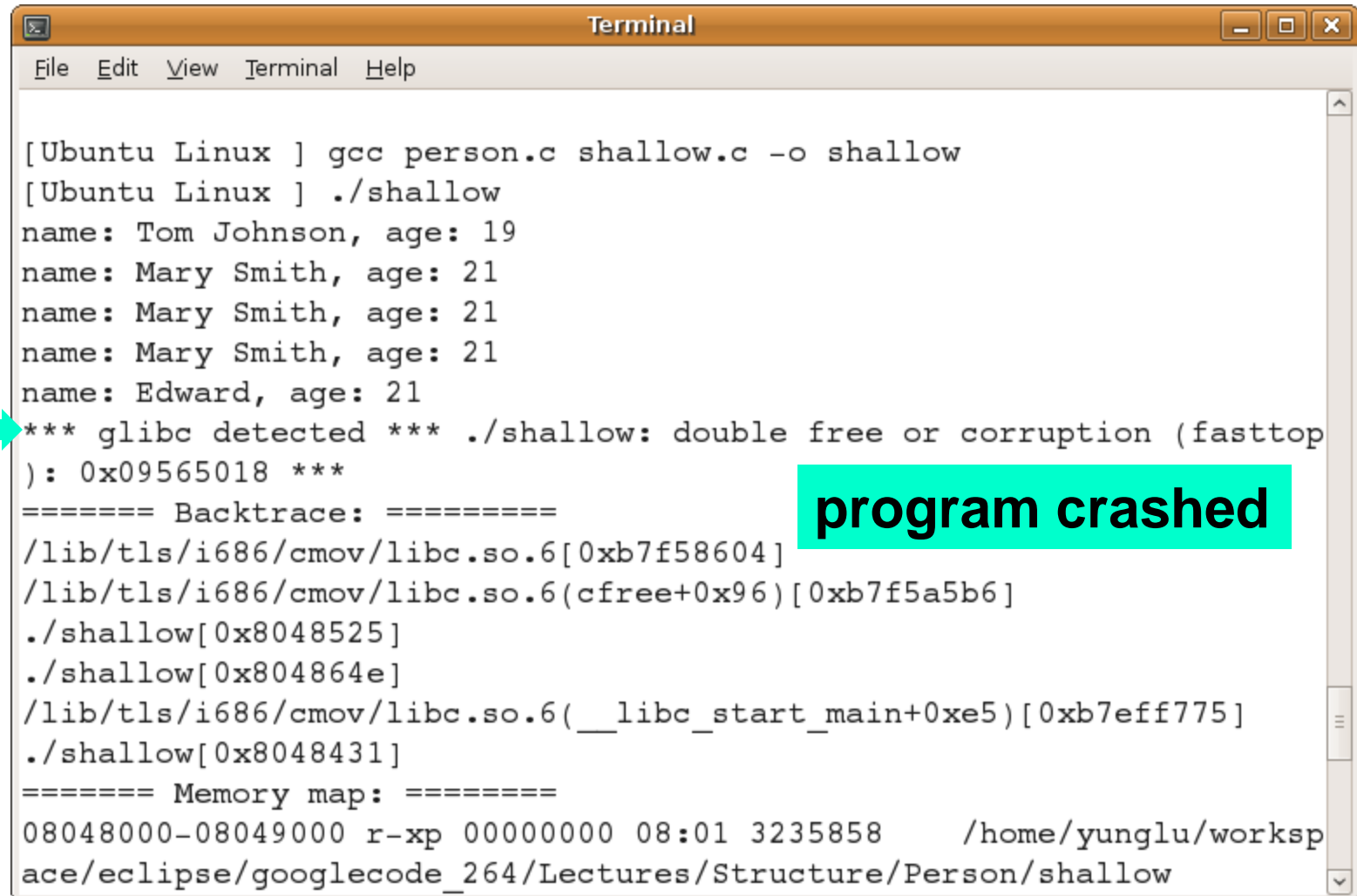
```
Person p2 = ...
```

```
p1 = p2;
```

changing p1's name also changes p2's name

⇒ different from the behaviors of x or y

# We have another problem!



```
Terminal
File Edit View Terminal Help

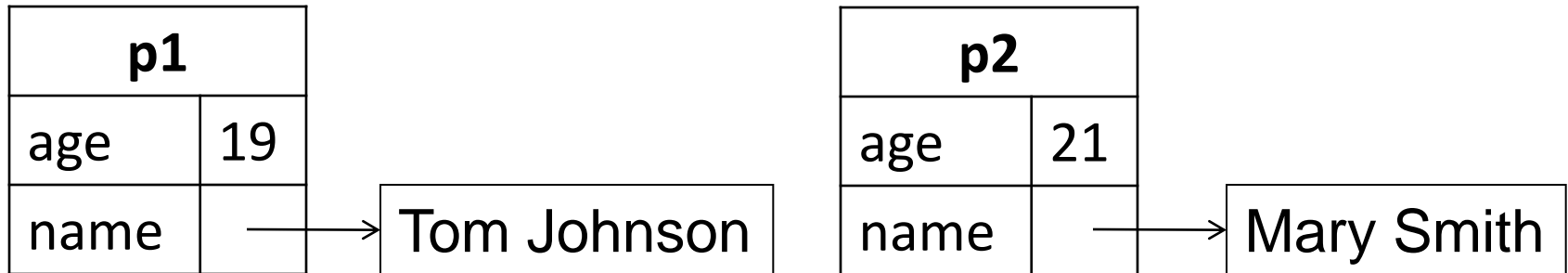
[Ubuntu Linux ] gcc person.c shallow.c -o shallow
[Ubuntu Linux ] ./shallow
name: Tom Johnson, age: 19
name: Mary Smith, age: 21
name: Mary Smith, age: 21
name: Mary Smith, age: 21
name: Edward, age: 21
*** glibc detected *** ./shallow: double free or corruption (fasttop
): 0x09565018 ***
===== Backtrace: =====
/lib/tls/i686/cmov/libc.so.6[0xb7f58604]
/lib/tls/i686/cmov/libc.so.6(cfree+0x96)[0xb7f5a5b6]
./shallow[0x8048525]
./shallow[0x804864e]
/lib/tls/i686/cmov/libc.so.6(__libc_start_main+0xe5)[0xb7eff775]
./shallow[0x8048431]
===== Memory map: =====
08048000-08049000 r-xp 00000000 08:01 3235858 /home/yunglu/worksp
ace/eclipse/googlecode_264/Lectures/Structure/Person/shallow
```

program crashed

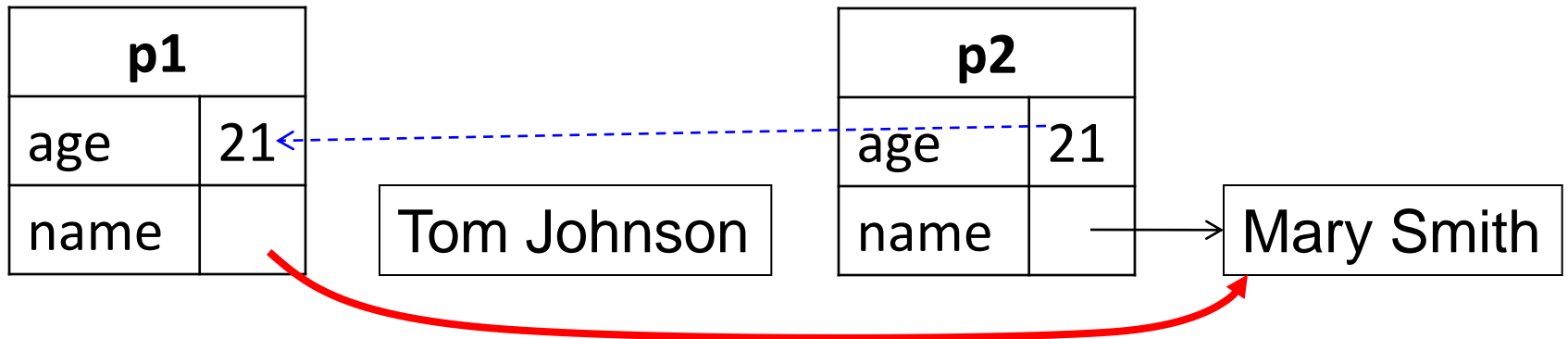
# What happened?

Person p1 = Person\_construct(19, "Tom Johnson");

Person p2 = Person\_construct(21, "Mary Smith");



p1 = p2; (copy the value of each attribute)



# Shallow Copy

What does `p1 = p2` do?

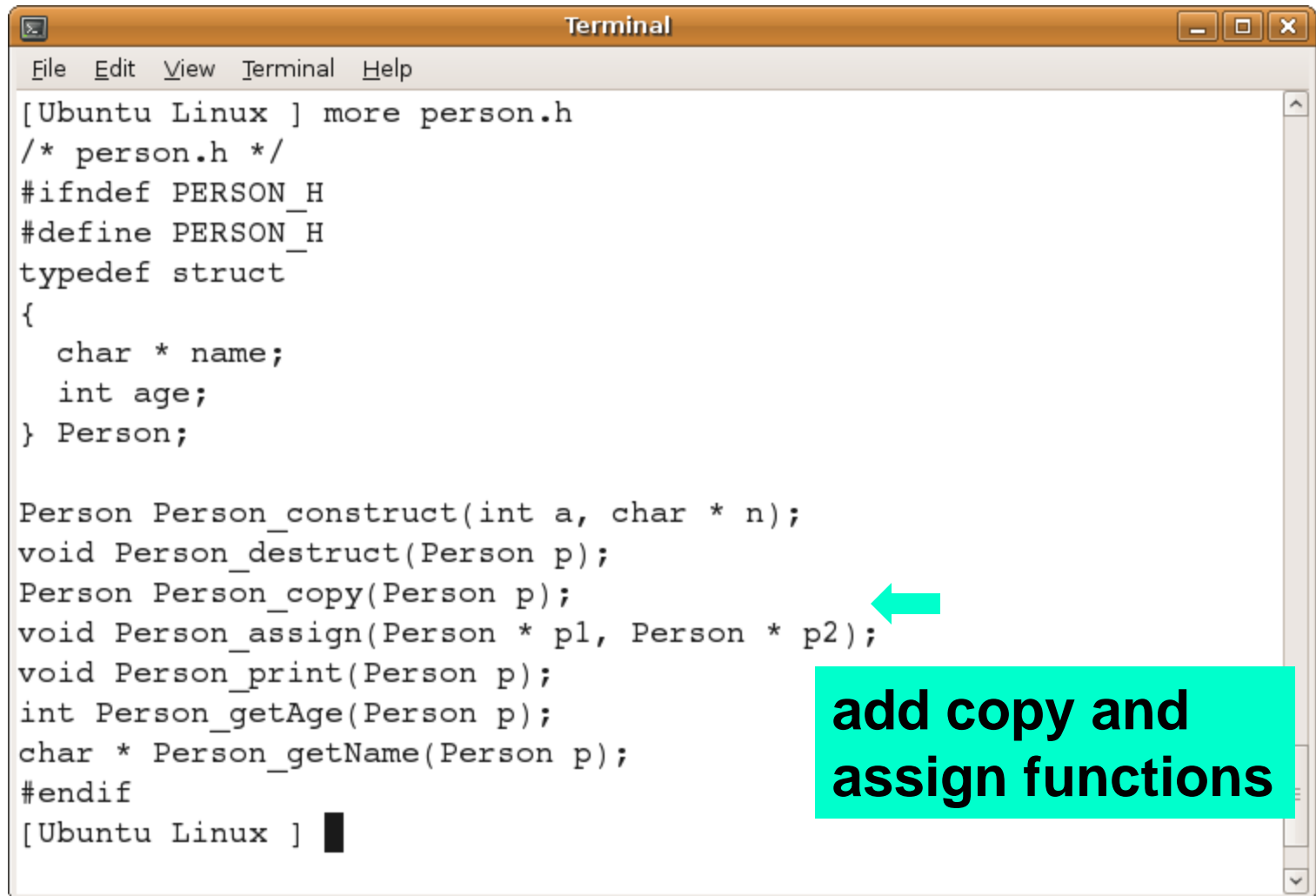
- **Copy p2's attribute values to p1's attributes**
  - If an attribute is a **pointer**  $\Rightarrow$  the value is an **address**
    - $\Rightarrow$  If two pointers have the same value, they **share** the same address
    - $\Rightarrow$  they point to the same **memory** location
  - p1's name is no longer accessible  $\Rightarrow$  memory leak
  - p1's and p2's names share the same memory  $\Rightarrow$  changing one changes the other
  - `Person_destruct(p1);`
  - `Person_destruct(p2);`
- delete the same memory twice**  
 **$\Rightarrow$  program crashes**

# **Solution: Deep Copy**

**Allocate Additional Memory  
Copy the Content (not Address)**



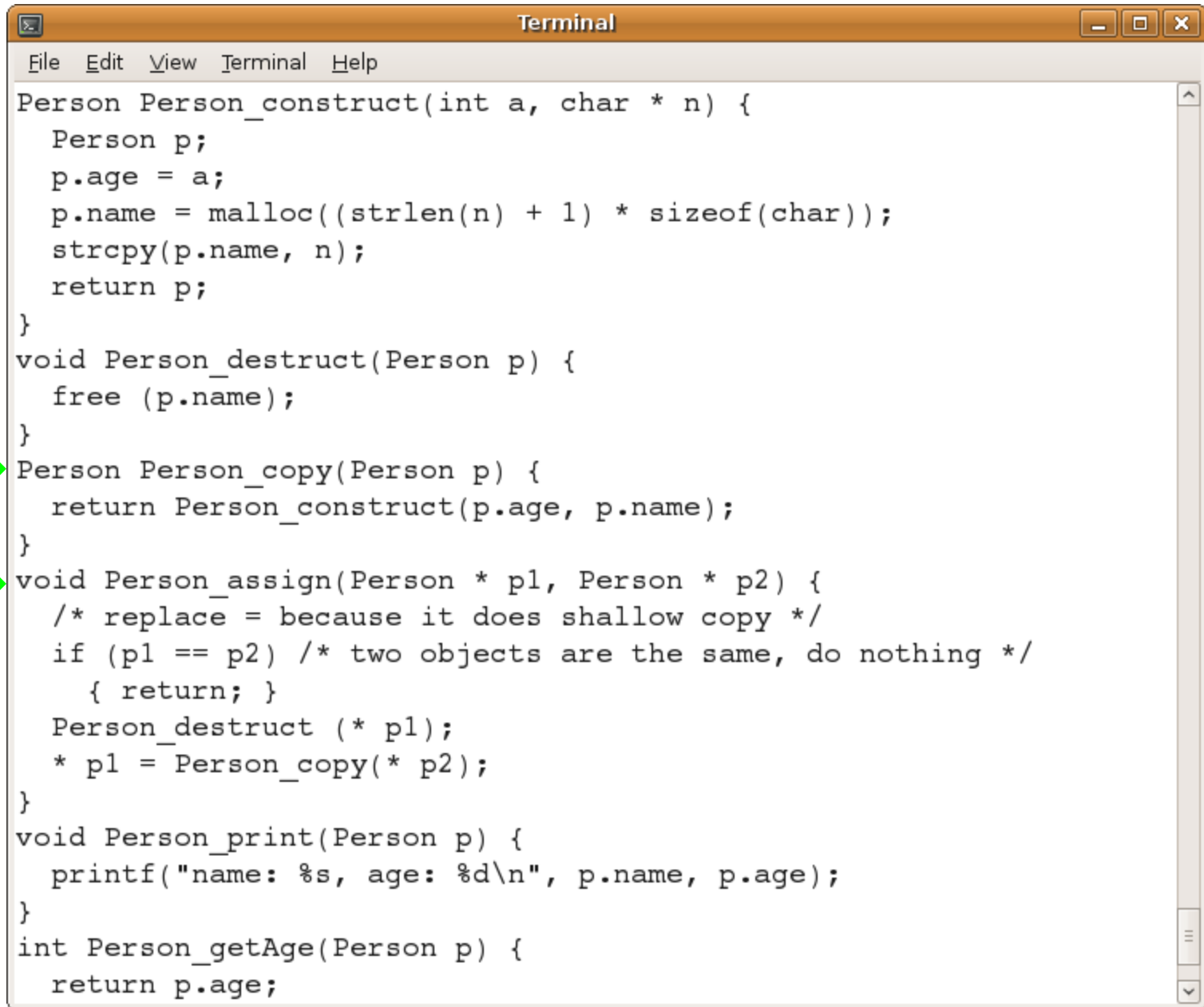
# new person.h

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Help). The terminal shows the output of the command "more person.h". The code displayed is the header file "person.h", which defines a "Person" struct and several functions. A red arrow points to the function declarations, and a red box highlights the text "add copy and assign functions".

```
[Ubuntu Linux ] more person.h
/* person.h */
#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);
Person Person_copy(Person p);
void Person_assign(Person * p1, Person * p2);
void Person_print(Person p);
int Person_getAge(Person p);
char * Person_getName(Person p);
#endif
[Ubuntu Linux ]
```

**add copy and  
assign functions**



A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Help) and standard window controls. It displays C code for a "Person" struct. The code includes functions for constructing, destructing, copying, assigning, printing, and getting the age of a person. Two green arrows point to the "Person\_copy" and "Person\_assign" function definitions.

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

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

→ Person Person_copy(Person p) {
    return Person_construct(p.age, p.name);
}

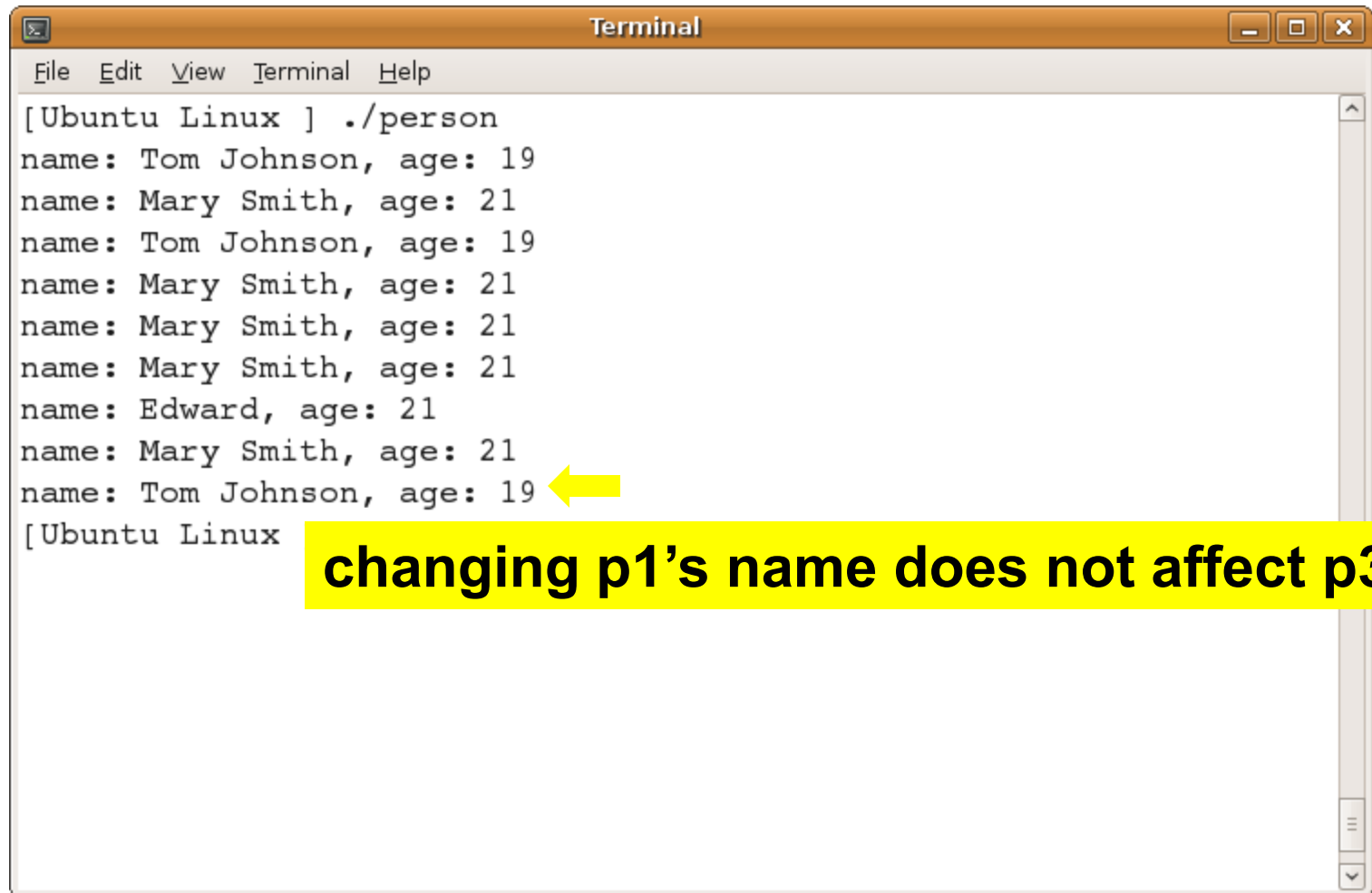
→ void Person_assign(Person * p1, Person * p2) {
    /* replace = because it does shallow copy */
    if (p1 == p2) /* two objects are the same, do nothing */
        { return; }
    Person_destruct (* p1);
    * p1 = Person_copy(* p2);
}

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

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

```
Terminal
File Edit View Terminal Help
[Ubuntu Linux ] more main.c
/* main.c */
#include <stdio.h>
#include <string.h>
#include "person.h"
int main(int argc, char * argv[])
{
    Person p1 = Person_construct(19, "Tom Johnson");
    Person p2 = Person_construct(21, "Mary Smith");
    Person p3;
    p3 = Person_copy(p1);
    Person_print(p1);
    Person_print(p2);
    Person_print(p3);
    Person_assign(& p1, & p2); /* notice & */
    Person_print(p1);
    Person_print(p2);
    Person_assign(& p1, & p1); /* source = destination */
    Person_print(p1);
    strcpy(p1.name, "Edward");
    Person_print(p1);
    Person_print(p2);
    Person_print(p3);
    Person_destruct(p1);
    Person_destruct(p2);
    Person_destruct(p3);
    return 0;
}
```

# execution



A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Help). The output shows a program running in an Ubuntu Linux environment. It prints a series of names and ages. A yellow arrow points to the line "name: Tom Johnson, age: 19", which is the last line of output before the prompt "[Ubuntu Linux ]".

```
[Ubuntu Linux ] ./person
name: Tom Johnson, age: 19
name: Mary Smith, age: 21
name: Tom Johnson, age: 19
name: Mary Smith, age: 21
name: Mary Smith, age: 21
name: Mary Smith, age: 21
name: Edward, age: 21
name: Mary Smith, age: 21
name: Tom Johnson, age: 19
[Ubuntu Linux ]
```

**changing p1's name does not affect p3**