Deep Copy

Yung-Hsiang Lu

Shallow Copy and the Problems

p1 = p2

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;
}
```

changing p1 changes p2, why?

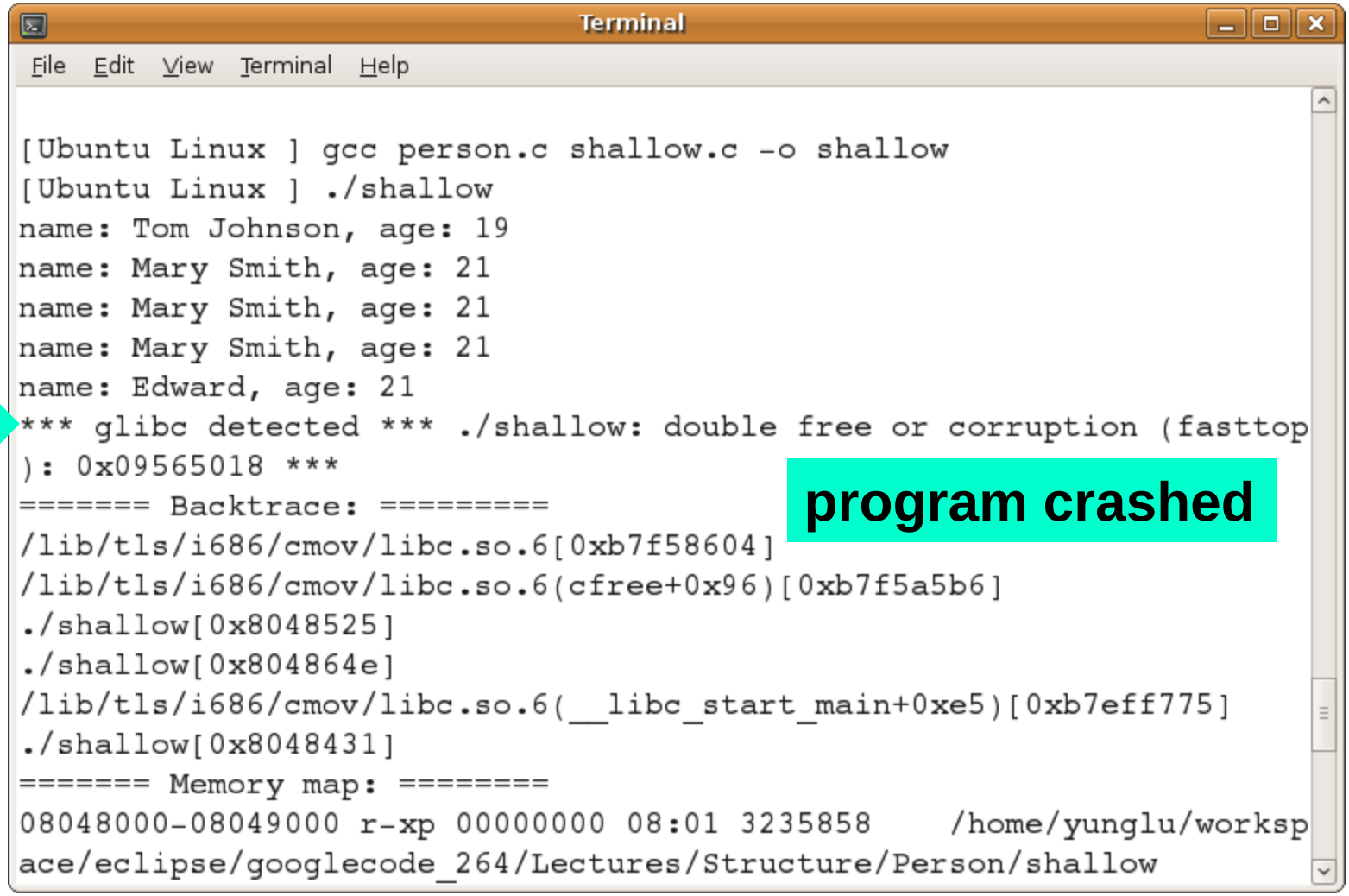
Variable Assignments

```
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

$\Rightarrow$ different from the behaviors of x or y

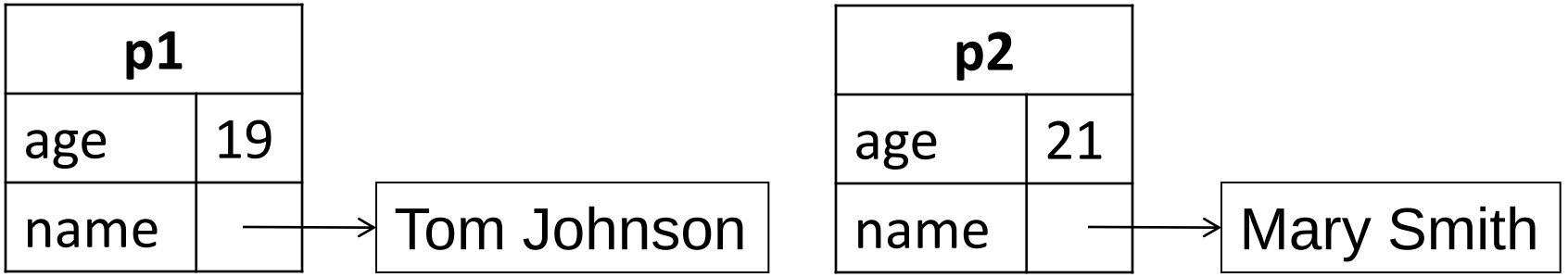
We have another problem!



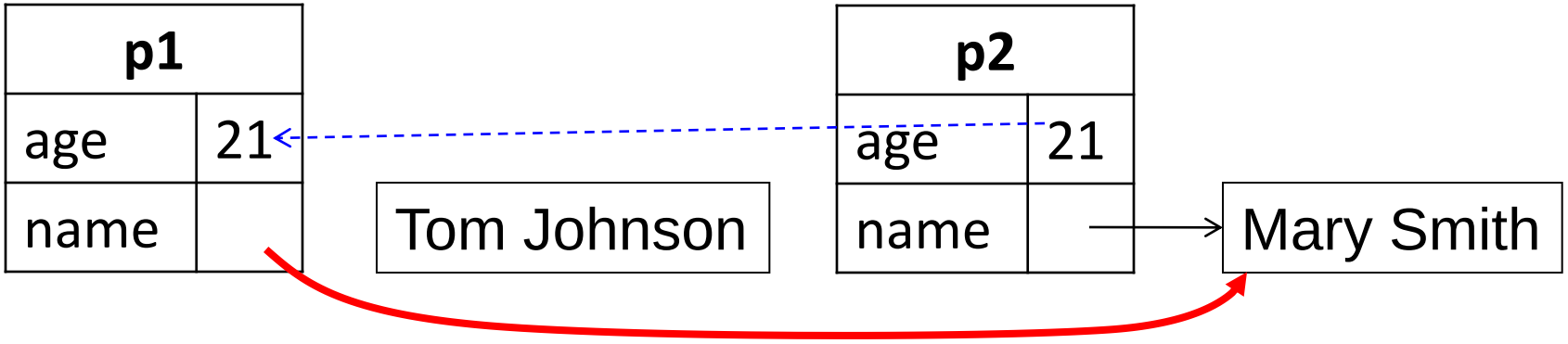
What happened?

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?

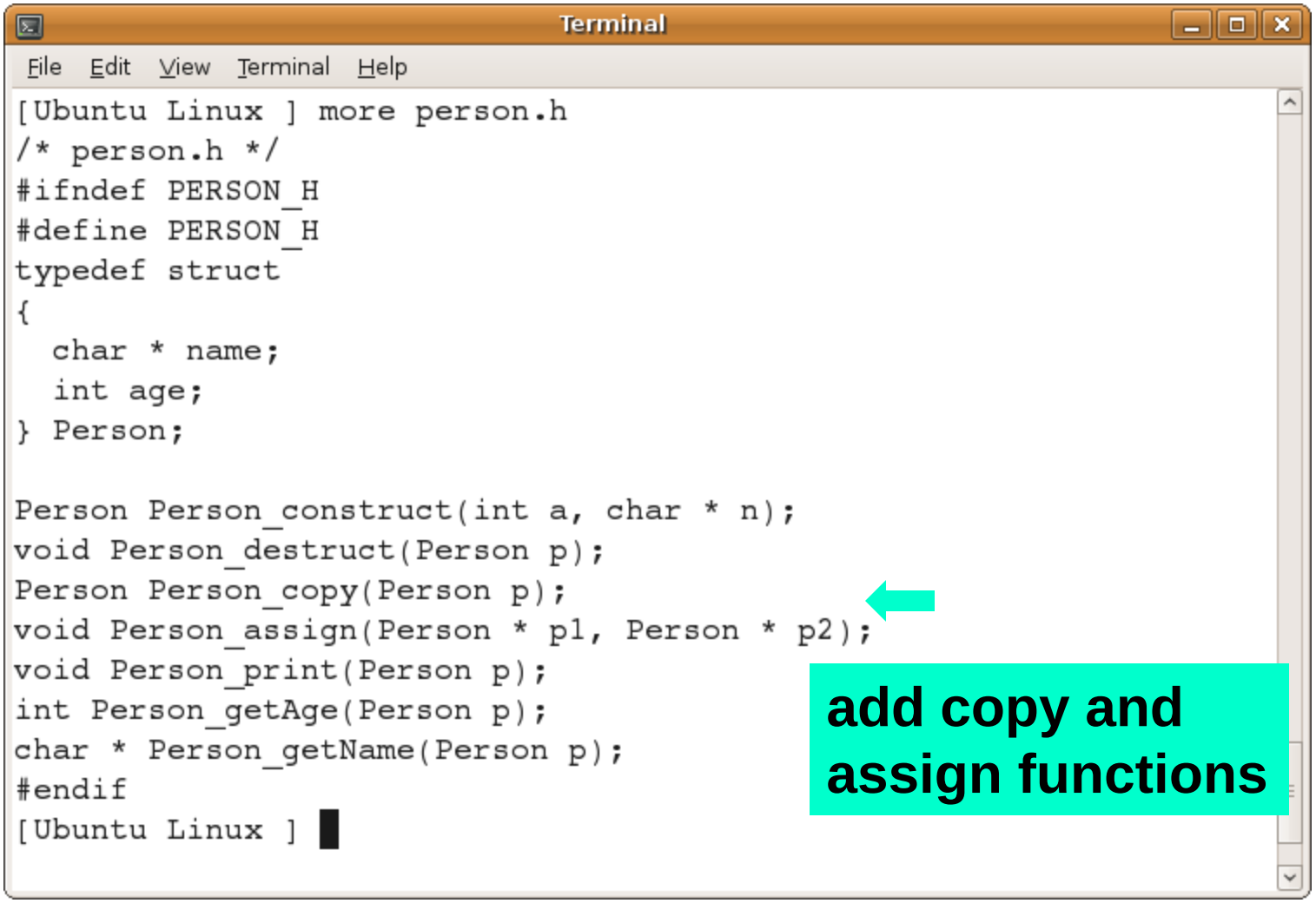
- **Copy p2's attribute values to p1's attributes**
- If an attribute is a **pointer** ⇒ the value is an **address**
  - ⇒ If two pointers have the same value, they **share** the same address
  - ⇒ they point to the same **memory** location
- p1's name is no longer accessible ⇒ memory leak
- p1's and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

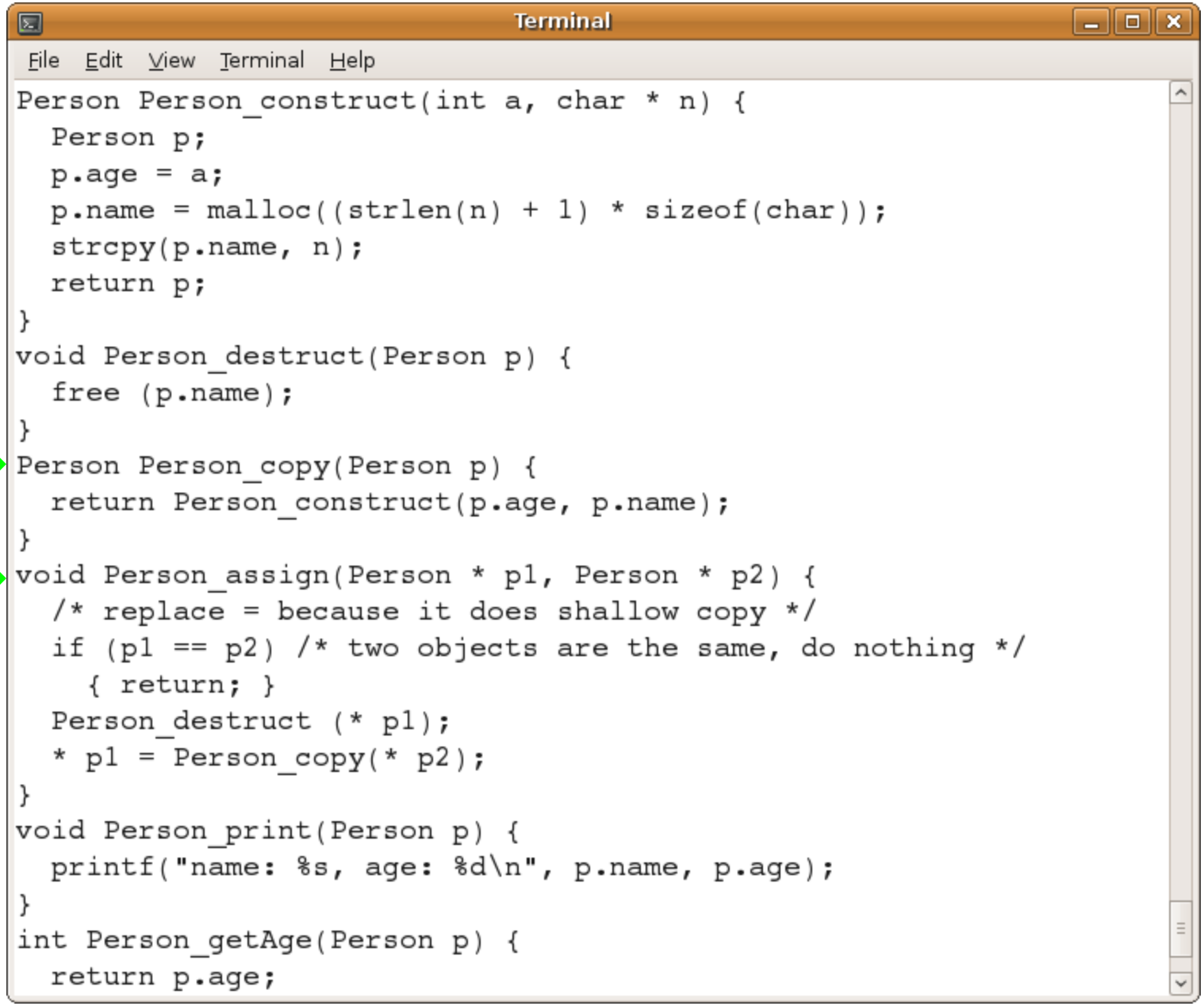
delete the same memory twice ⇒ program crashes

Solution: Deep Copy

Allocate Additional Memory
Copy the Content (not Address)

new person.h


Terminal
File Edit View Terminal Help
[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

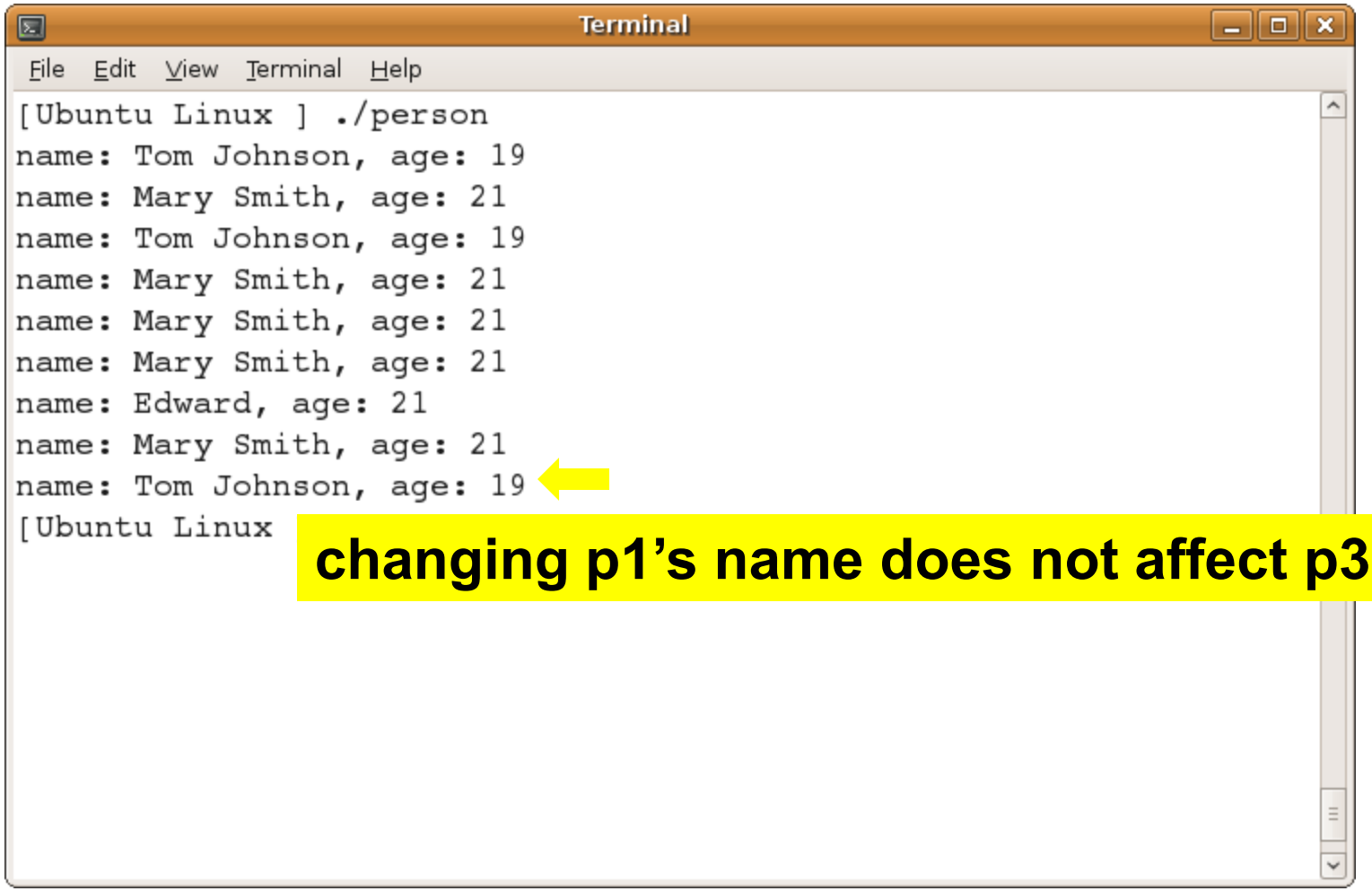

Terminal
File Edit View Terminal Help
Person Person_construct(int a, char * n) {
Person p;
p.age = a;
p.name = malloc((strlen(n) + 1) * sizeof(char));
strcpy(p.name, n);
return p;
}
void Person_destruct(Person p) {
free (p.name);
}
Person Person_copy(Person p) {
return Person_construct(p.age, p.name);
}
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


Terminal
File Edit View Terminal Help
[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