Flow in get_new_id

```
int get_new_id() {
    int id = getpid();
    return id;
}
```

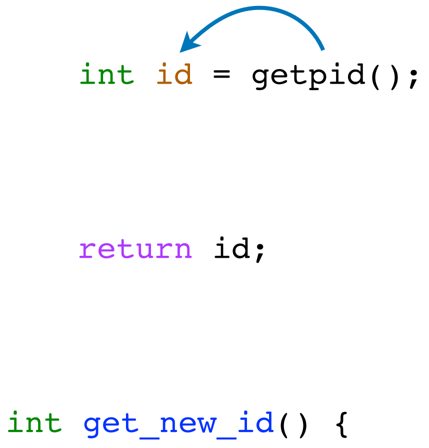
```
    int id = getpid();
```

```
    return id;
```

```
int get_new_id() {
```

Flow in get_new_id

```
int get_new_id() {
    int id = getpid();
    return id;
}
```



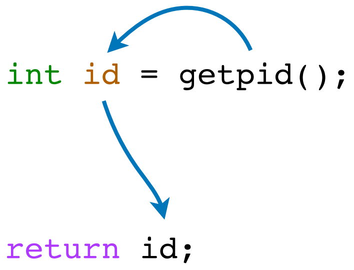
```
int id = getpid();
```

```
return id;
```

```
int get_new_id() {
```

Flow in get_new_id

```
int get_new_id() {
    int id = getpid();
    return id;
}
```



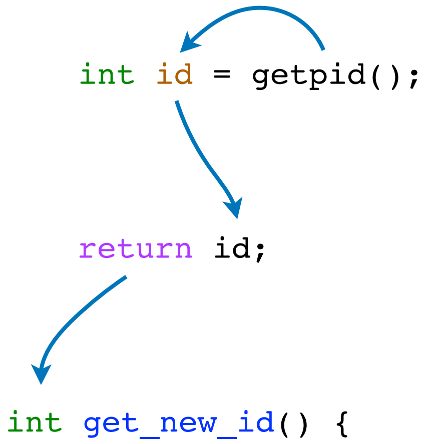
```
int id = getpid();
```

```
return id;
```

```
int get_new_id() {
```

Flow in get_new_id

```
int get_new_id() {
    int id = getpid();
    return id;
}
```



```
int id = getpid();

return id;

int get_new_id() {
```

Flow in get_user_info

```c
char* get_user_info() {
#define BUFSIZE 1024
    char* buf = (char*) malloc(BUFSIZE * sizeof(char));
    int count;
    // Disable buffering to avoid need for fflush
    // after printf().
    setbuf( stdout, NULL );
    printf("*** Welcome to sql injection ***\n");
    printf("Please enter name: ");
    count = read(STDIN_FILENO, buf, BUFSIZE);
    if (count <= 0) abort();
    /* strip trailing whitespace */
    while (count && isspace(buf[count-1])) {
        buf[count-1] = 0; --count;
    }
    return buf;
}
```

Agent Smith

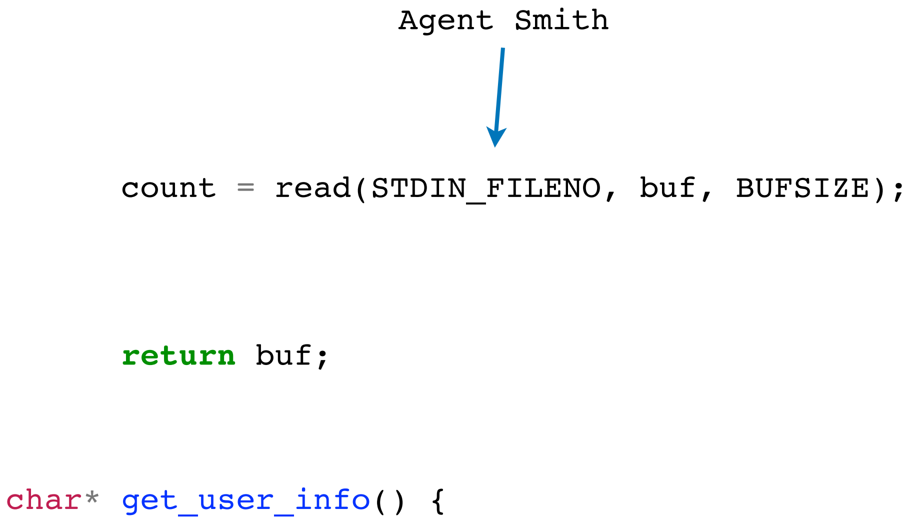
```c
count = read(STDIN_FILENO, buf, BUFSIZE);
```

```c
return buf;
```

```c
char* get_user_info() {
```

Flow in get_user_info

```c
char* get_user_info() {
#define BUFSIZE 1024
    char* buf = (char*) malloc(BUFSIZE * sizeof(char));
    int count;
    // Disable buffering to avoid need for fflush
    // after printf().
    setbuf( stdout, NULL );
    printf("*** Welcome to sql injection ***\n");
    printf("Please enter name: ");
    count = read(STDIN_FILENO, buf, BUFSIZE);
    if (count <= 0) abort();
    /* strip trailing whitespace */
    while (count && isspace(buf[count-1])) {
        buf[count-1] = 0; --count;
    }
    return buf;
}
```

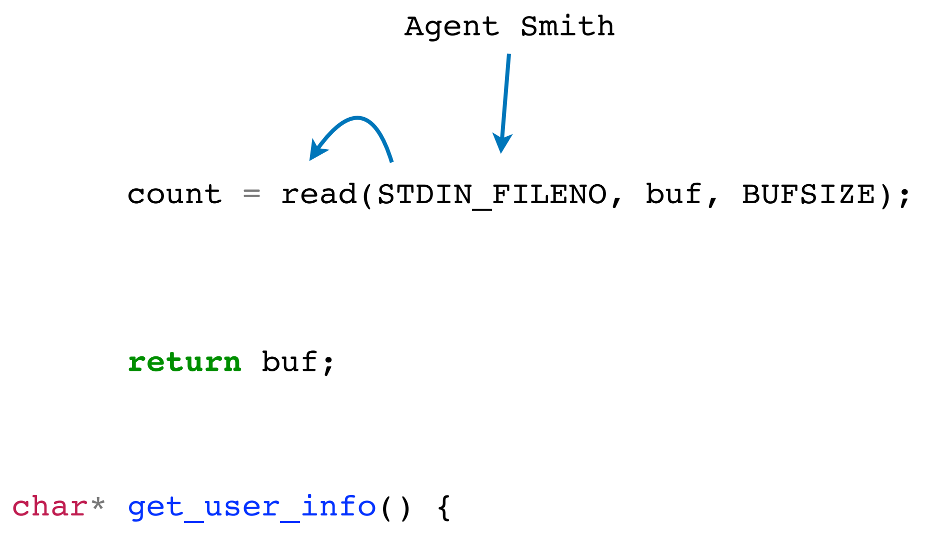


Agent Smith

```c
    count = read(STDIN_FILENO, buf, BUFSIZE);


    return buf;


char* get_user_info() {
```

Flow in get_user_info

```c
char* get_user_info() {
#define BUFSIZE 1024
    char* buf = (char*) malloc(BUFSIZE * sizeof(char));
    int count;
    // Disable buffering to avoid need for fflush
    // after printf().
    setbuf( stdout, NULL );
    printf("*** Welcome to sql injection ***\n");
    printf("Please enter name: ");
    count = read(STDIN_FILENO, buf, BUFSIZE);
    if (count <= 0) abort();
    /* strip trailing whitespace */
    while (count && isspace(buf[count-1])) {
        buf[count-1] = 0; --count;
    }
    return buf;
}
```

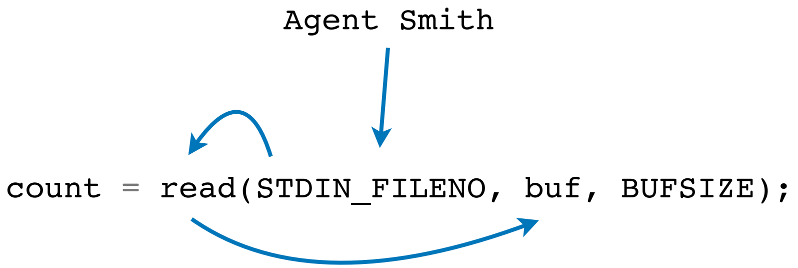


Agent Smith

```c
count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```

Flow in get_user_info

```c
char* get_user_info() {
#define BUFSIZE 1024
    char* buf = (char*) malloc(BUFSIZE * sizeof(char));
    int count;
    // Disable buffering to avoid need for fflush
    // after printf().
    setbuf( stdout, NULL );
    printf("*** Welcome to sql injection ***\n");
    printf("Please enter name: ");
    count = read(STDIN_FILENO, buf, BUFSIZE);
    if (count <= 0) abort();
    /* strip trailing whitespace */
    while (count && isspace(buf[count-1])) {
        buf[count-1] = 0; --count;
    }
    return buf;
}
```



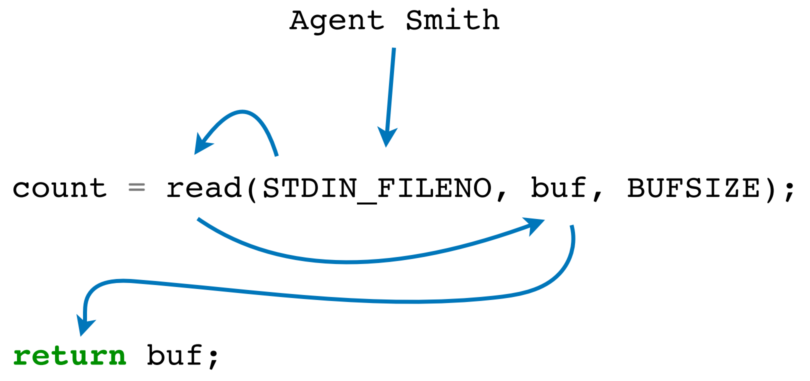
Agent Smith

```c
count = read(STDIN_FILENO, buf, BUFSIZE);
```

```c
  return buf;


char* get_user_info() {
```

Flow in get_user_info

```
char* get_user_info() {
#define BUFSIZE 1024
    char* buf = (char*) malloc(BUFSIZE * sizeof(char));
    int count;
    // Disable buffering to avoid need for fflush
    // after printf().
    setbuf( stdout, NULL );
    printf("*** Welcome to sql injection ***\n");
    printf("Please enter name: ");
    count = read(STDIN_FILENO, buf, BUFSIZE);
    if (count <= 0) abort();
    /* strip trailing whitespace */
    while (count && isspace(buf[count-1])) {
        buf[count-1] = 0; --count;
    }
    return buf;
}
```



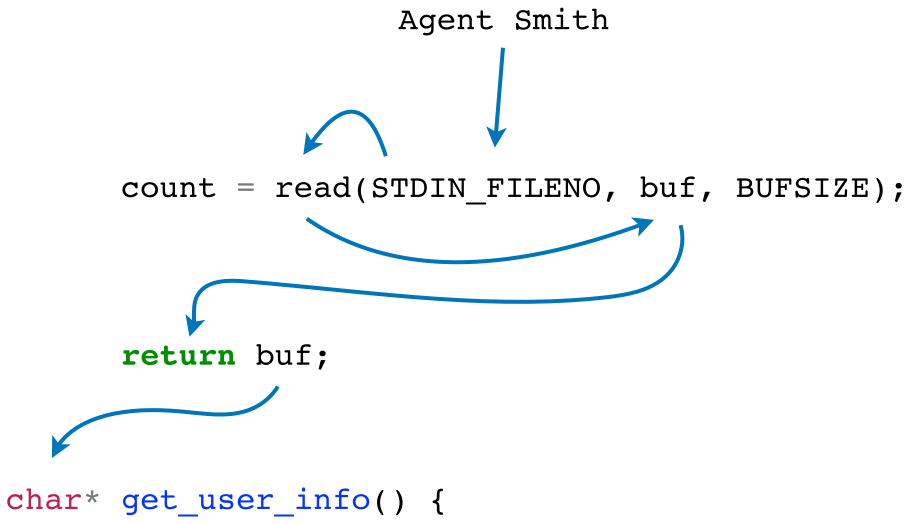
Agent Smith

```
count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;
```

```
char* get_user_info() {
```

Flow in get_user_info

```
char* get_user_info() {
#define BUFSIZE 1024
    char* buf = (char*) malloc(BUFSIZE * sizeof(char));
    int count;
    // Disable buffering to avoid need for fflush
    // after printf().
    setbuf( stdout, NULL );
    printf("*** Welcome to sql injection ***\n");
    printf("Please enter name: ");
    count = read(STDIN_FILENO, buf, BUFSIZE);
    if (count <= 0) abort();
    /* strip trailing whitespace */
    while (count && isspace(buf[count-1])) {
        buf[count-1] = 0; --count;
    }
    return buf;
}
```



Agent Smith

```
count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```

Flow in write_info

```
void write_info(int id, char* info) {
    sqlite3 *db;
    int rc;
    int bufsize = 1024;
    char *zErrMsg = 0;
    char query[bufsize];

    /* open db */
    rc = sqlite3_open("users.sqlite", &db);
    abort_on_error(rc, db);

    /* Format query */
    snprintf(query, bufsize,
             "INSERT INTO users VALUES (%d, '%s')",
             id, info);
    write_log("query: %s\n", query);

    /* Write info */
    rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
    abort_on_exec_error(rc, db, zErrMsg);

    sqlite3_close(db);
}
```

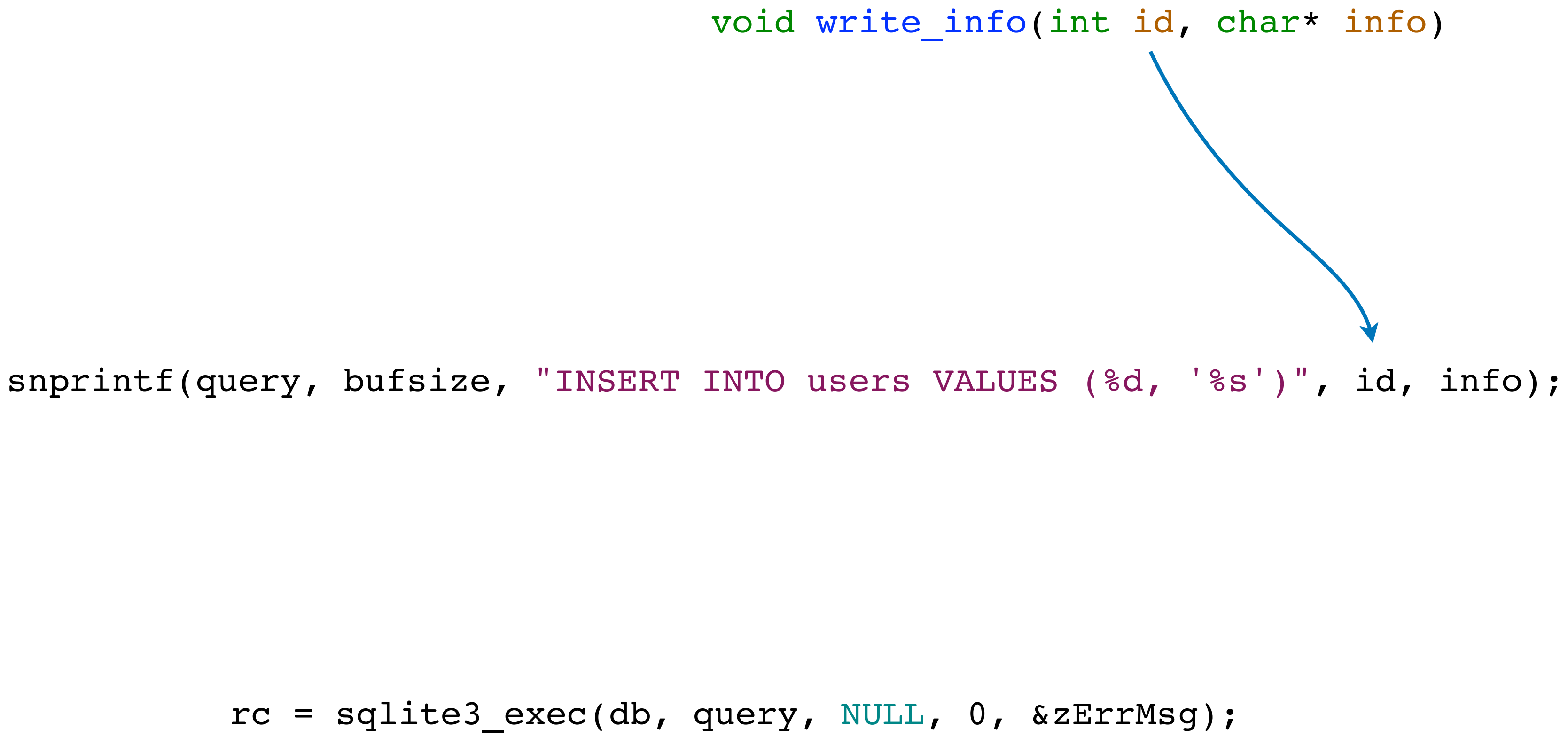
```
void write_info(int id, char* info)
```

```
snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
```

```
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

Flow in write_info

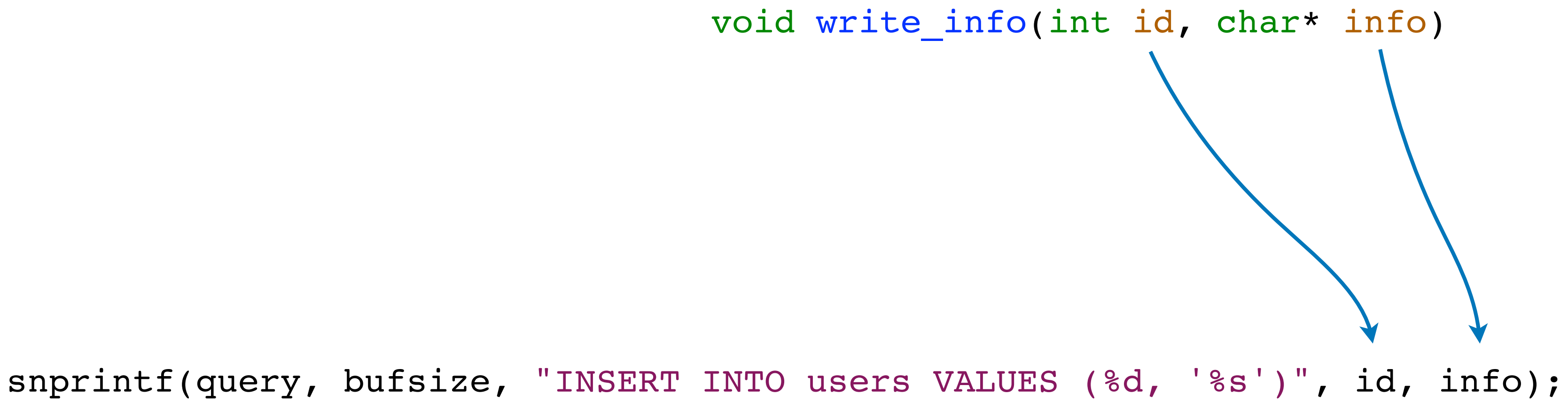
```
void write_info(int id, char* info) {
    sqlite3 *db;
    int rc;
    int bufsize = 1024;
    char *zErrMsg = 0;
    char query[bufsize];

    /* open db */
    rc = sqlite3_open("users.sqlite", &db);
    abort_on_error(rc, db);

    /* Format query */
    snprintf(query, bufsize,
             "INSERT INTO users VALUES (%d, '%s')",
             id, info);
    write_log("query: %s\n", query);

    /* Write info */
    rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
    abort_on_exec_error(rc, db, zErrMsg);

    sqlite3_close(db);
}
```



```
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

Flow in write_info

```c
void write_info(int id, char* info) {
    sqlite3 *db;
    int rc;
    int bufsize = 1024;
    char *zErrMsg = 0;
    char query[bufsize];

    /* open db */
    rc = sqlite3_open("users.sqlite", &db);
    abort_on_error(rc, db);

    /* Format query */
    snprintf(query, bufsize,
             "INSERT INTO users VALUES (%d, '%s')",
             id, info);
    write_log("query: %s\n", query);

    /* Write info */
    rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
    abort_on_exec_error(rc, db, zErrMsg);

    sqlite3_close(db);
}
```



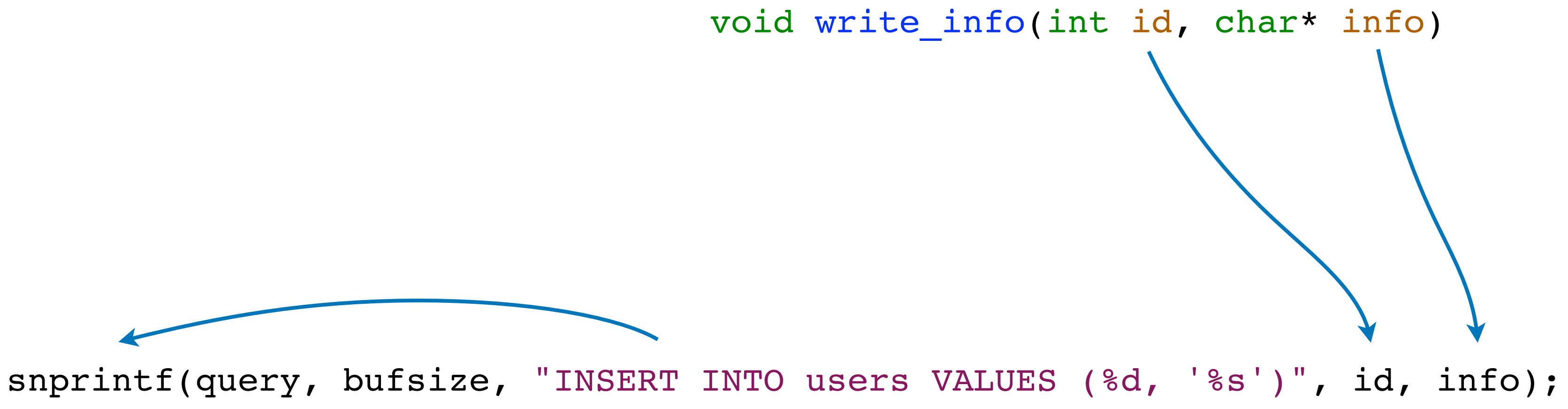
```c
void write_info(int id, char* info)
```

```c
snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
```

```c
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

Flow in write_info

```c
void write_info(int id, char* info) {
    sqlite3 *db;
    int rc;
    int bufsize = 1024;
    char *zErrMsg = 0;
    char query[bufsize];

    /* open db */
    rc = sqlite3_open("users.sqlite", &db);
    abort_on_error(rc, db);

    /* Format query */
    snprintf(query, bufsize,
             "INSERT INTO users VALUES (%d, '%s')",
             id, info);
    write_log("query: %s\n", query);

    /* Write info */
    rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
    abort_on_exec_error(rc, db, zErrMsg);

    sqlite3_close(db);
}
```

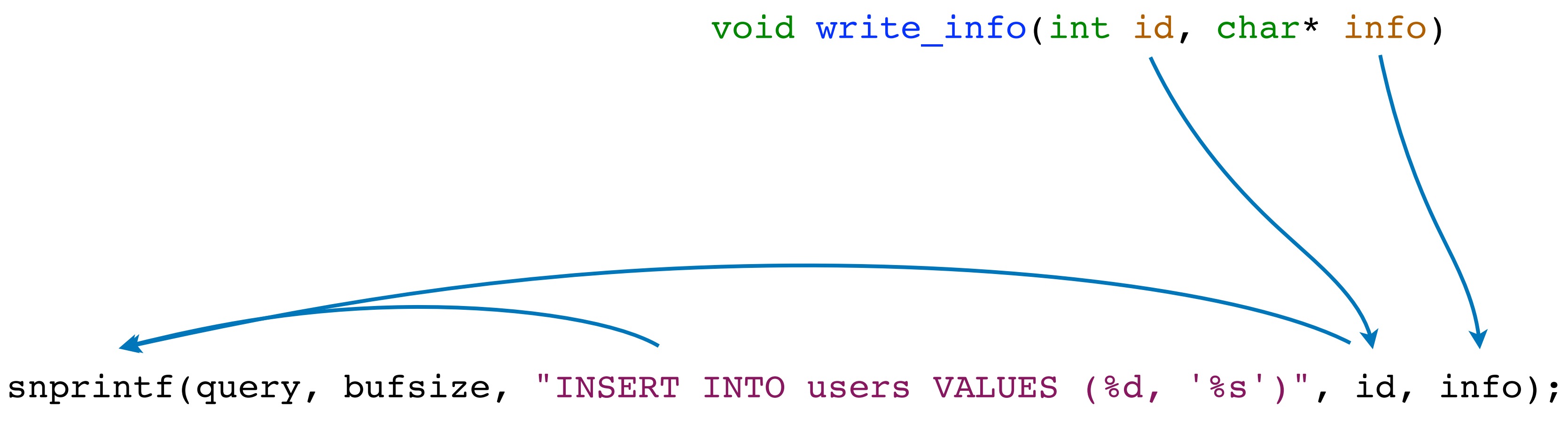


```c
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
```

```c
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

Flow in write_info

```
void write_info(int id, char* info) {
    sqlite3 *db;
    int rc;
    int bufsize = 1024;
    char *zErrMsg = 0;
    char query[bufsize];

    /* open db */
    rc = sqlite3_open("users.sqlite", &db);
    abort_on_error(rc, db);

    /* Format query */
    snprintf(query, bufsize,
             "INSERT INTO users VALUES (%d, '%s')",
             id, info);
    write_log("query: %s\n", query);

    /* Write info */
    rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
    abort_on_exec_error(rc, db, zErrMsg);

    sqlite3_close(db);
}
```



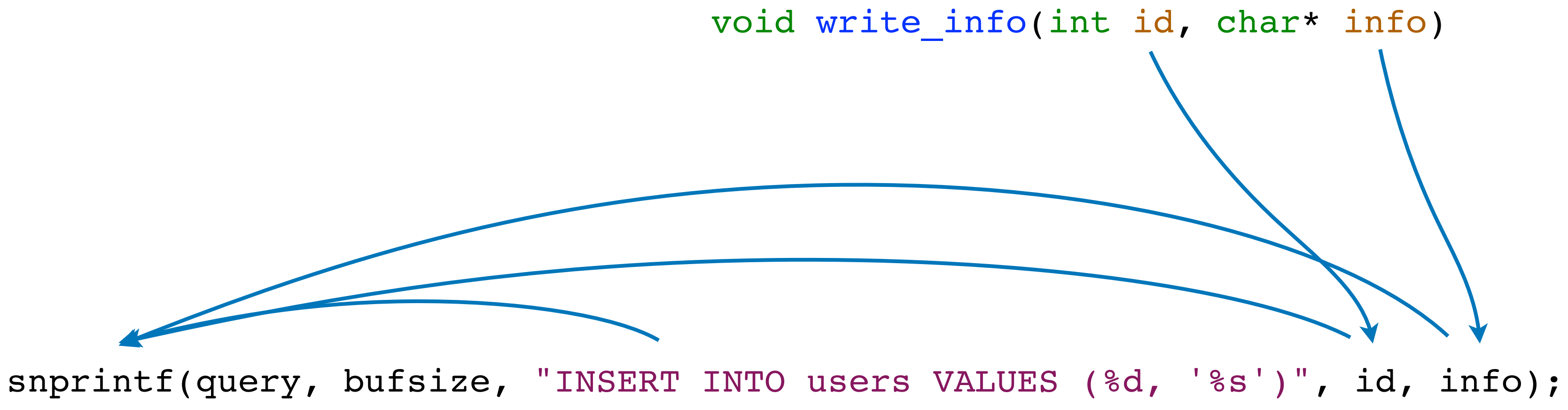
```
void write_info(int id, char* info)
```

```
snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
```

```
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

Flow in write_info

```
void write_info(int id, char* info) {
    sqlite3 *db;
    int rc;
    int bufsize = 1024;
    char *zErrMsg = 0;
    char query[bufsize];

    /* open db */
    rc = sqlite3_open("users.sqlite", &db);
    abort_on_error(rc, db);

    /* Format query */
    snprintf(query, bufsize,
             "INSERT INTO users VALUES (%d, '%s')",
             id, info);
    write_log("query: %s\n", query);

    /* Write info */
    rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
    abort_on_exec_error(rc, db, zErrMsg);

    sqlite3_close(db);
}
```



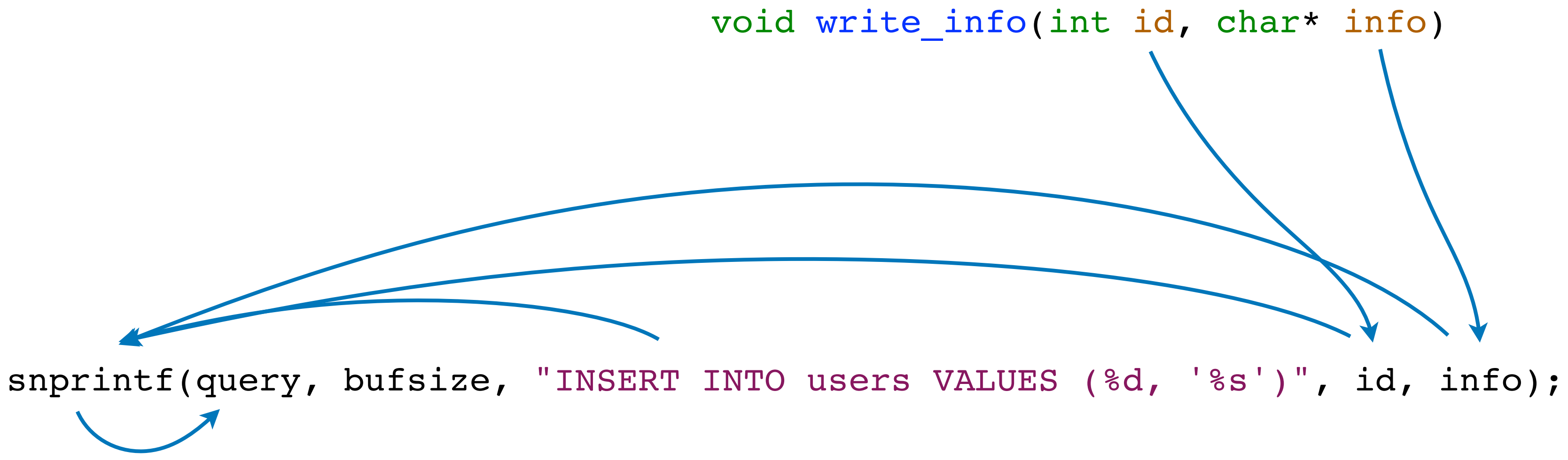
```
void write_info(int id, char* info)
```

```
snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
```

```
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

Flow in write_info

```
void write_info(int id, char* info) {
    sqlite3 *db;
    int rc;
    int bufsize = 1024;
    char *zErrMsg = 0;
    char query[bufsize];

    /* open db */
    rc = sqlite3_open("users.sqlite", &db);
    abort_on_error(rc, db);

    /* Format query */
    snprintf(query, bufsize,
             "INSERT INTO users VALUES (%d, '%s')",
             id, info);
    write_log("query: %s\n", query);

    /* Write info */
    rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
    abort_on_exec_error(rc, db, zErrMsg);

    sqlite3_close(db);
}
```

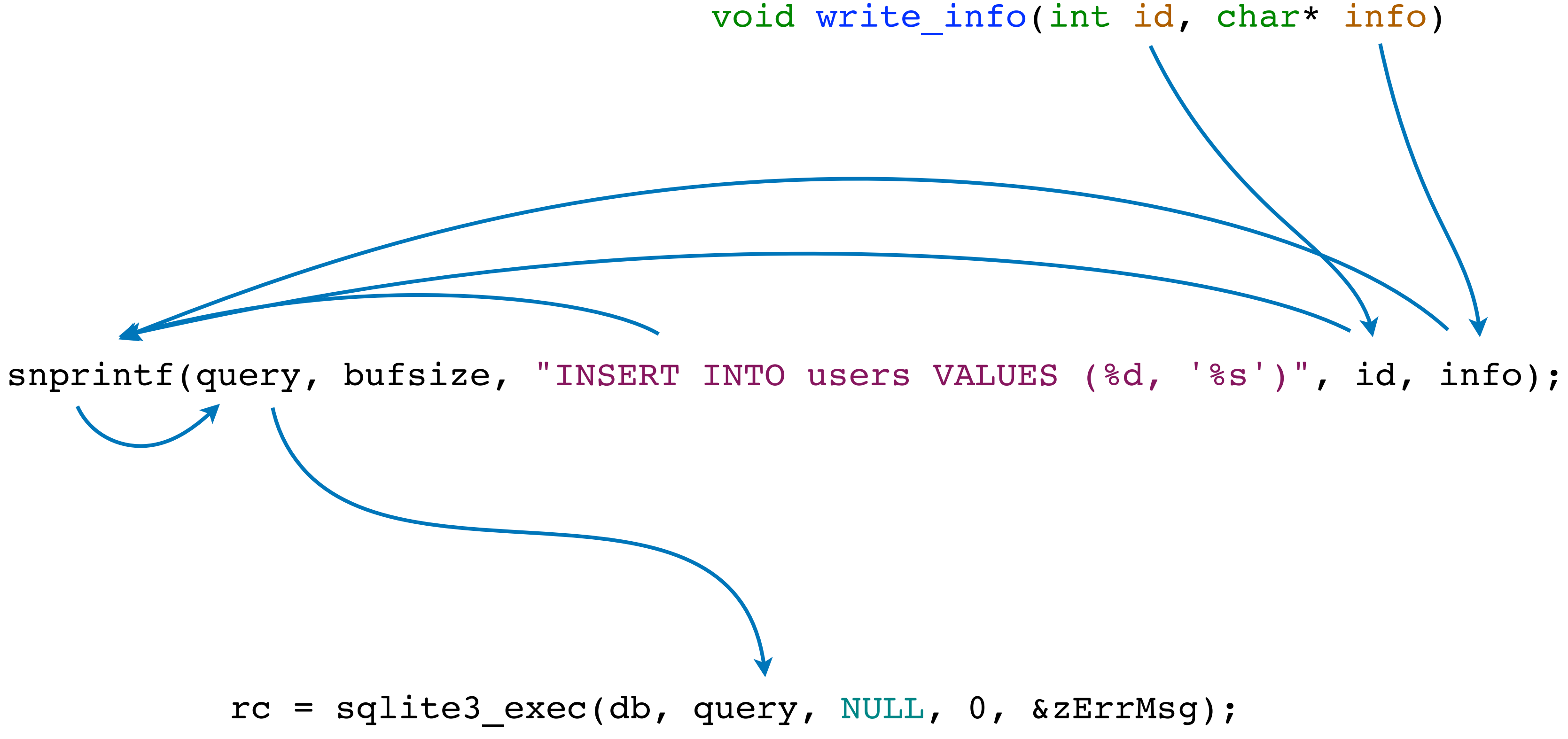


```
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
```

```
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

Flow in write_info

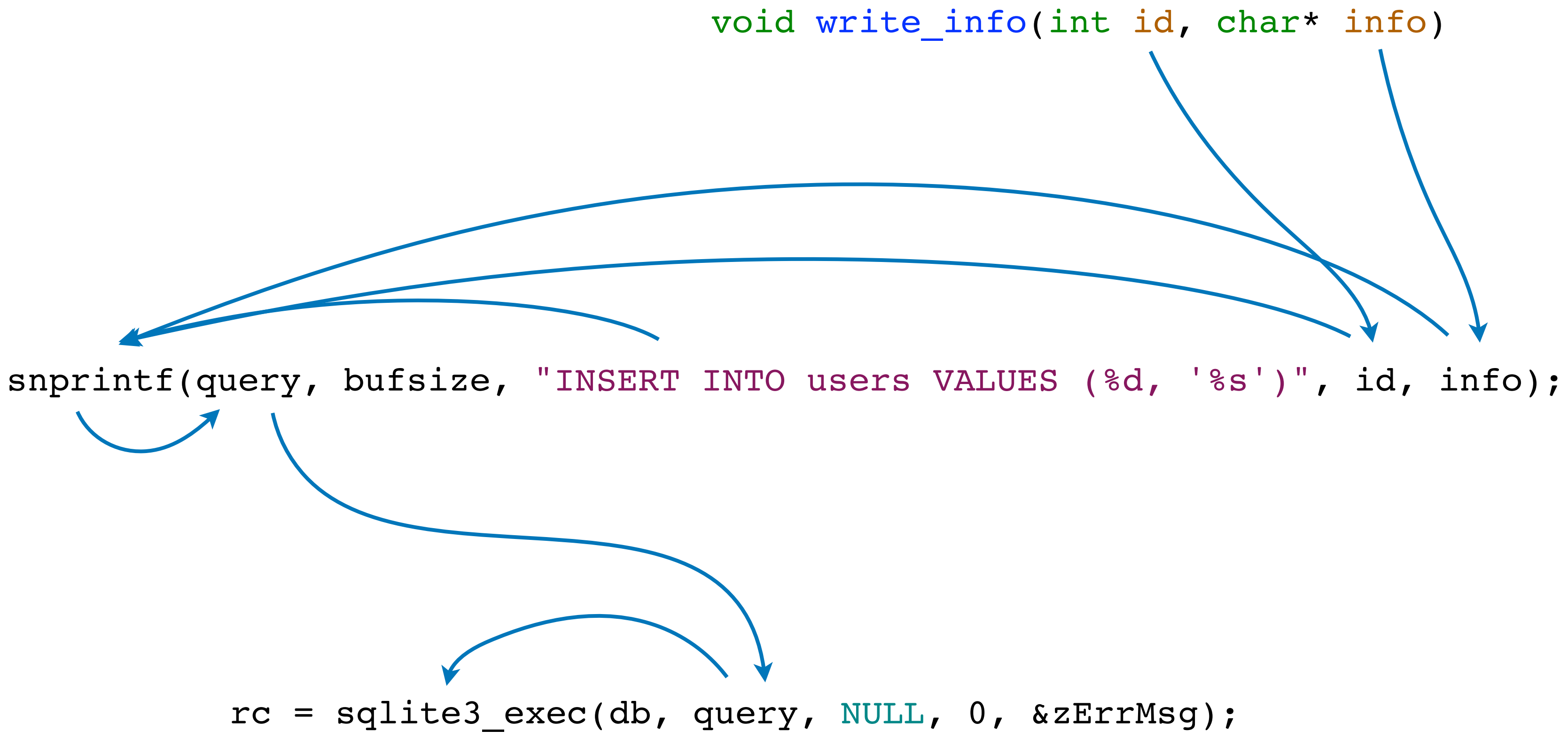
```c
void write_info(int id, char* info) {
    sqlite3 *db;
    int rc;
    int bufsize = 1024;
    char *zErrMsg = 0;
    char query[bufsize];

    /* open db */
    rc = sqlite3_open("users.sqlite", &db);
    abort_on_error(rc, db);

    /* Format query */
    snprintf(query, bufsize,
             "INSERT INTO users VALUES (%d, '%s')",
             id, info);
    write_log("query: %s\n", query);

    /* Write info */
    rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
    abort_on_exec_error(rc, db, zErrMsg);

    sqlite3_close(db);
}
```



```c
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

Flow in write_info

```
void write_info(int id, char* info) {
    sqlite3 *db;
    int rc;
    int bufsize = 1024;
    char *zErrMsg = 0;
    char query[bufsize];

    /* open db */
    rc = sqlite3_open("users.sqlite", &db);
    abort_on_error(rc, db);

    /* Format query */
    snprintf(query, bufsize,
             "INSERT INTO users VALUES (%d, '%s')",
             id, info);
    write_log("query: %s\n", query);

    /* Write info */
    rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
    abort_on_exec_error(rc, db, zErrMsg);

    sqlite3_close(db);
}
```



```
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

Flow in main

```
int main(int argc, char* argv[]) {
    char* info;
    int id;
    info = get_user_info();
    id = get_new_id();
    write_info(id, info);
}
```

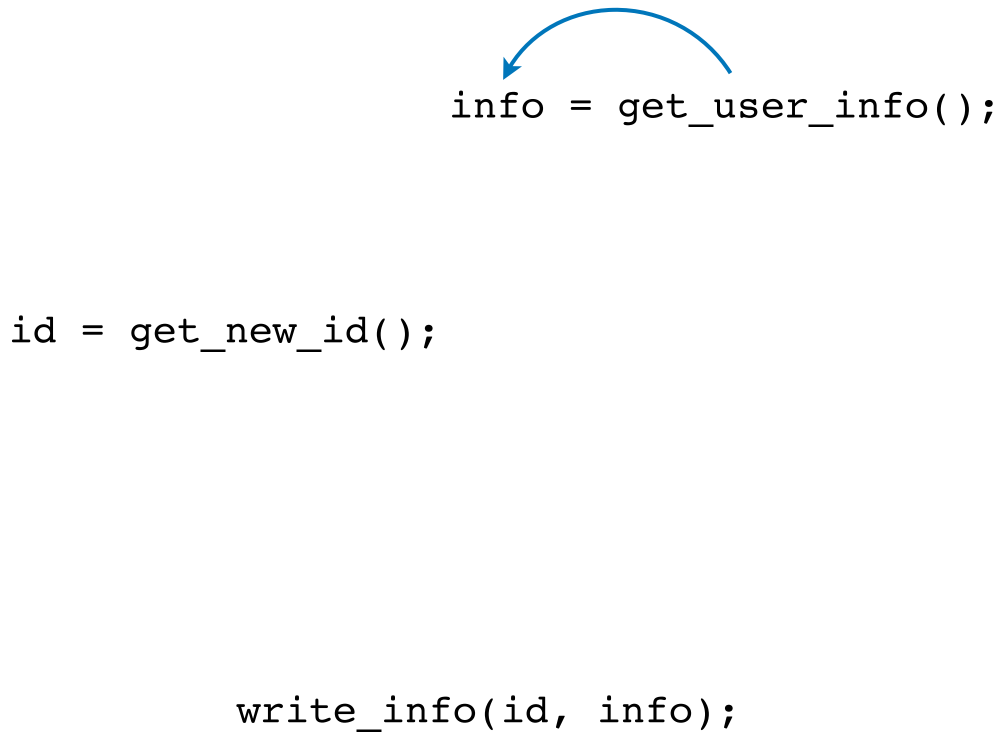
```
info = get_user_info();
```

```
id = get_new_id();
```

```
write_info(id, info);
```

Flow in main

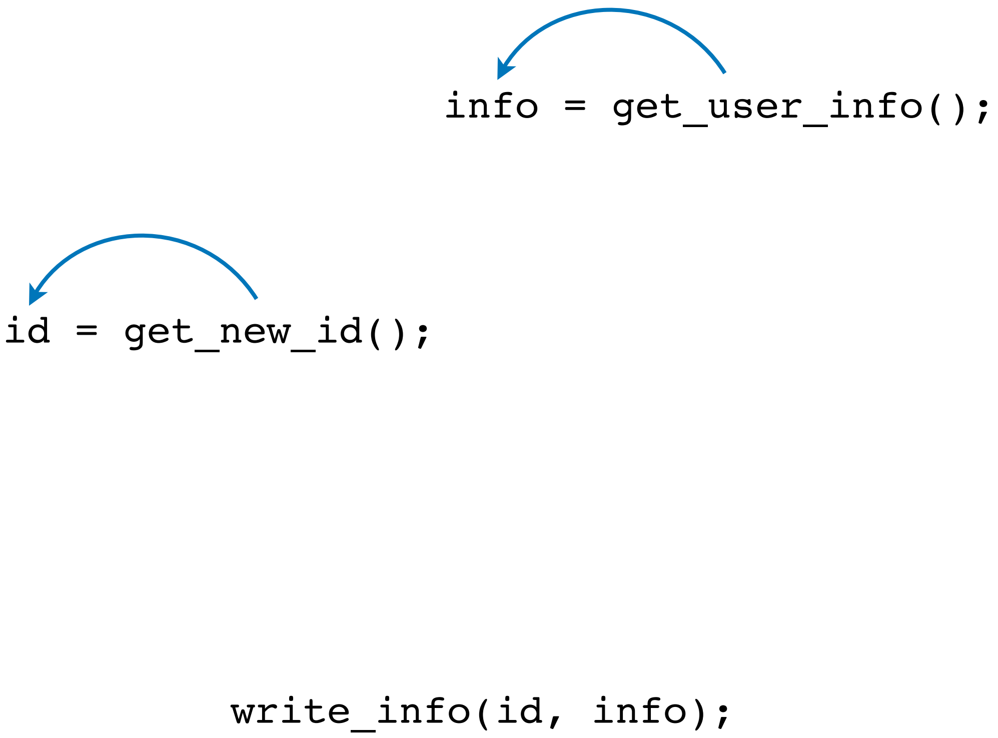
```
int main(int argc, char* argv[]) {

    char* info;

    int id;

    info = get_user_info();

    id = get_new_id();

    write_info(id, info);

}
```



```
info = get_user_info();

id = get_new_id();

write_info(id, info);
```

Flow in main

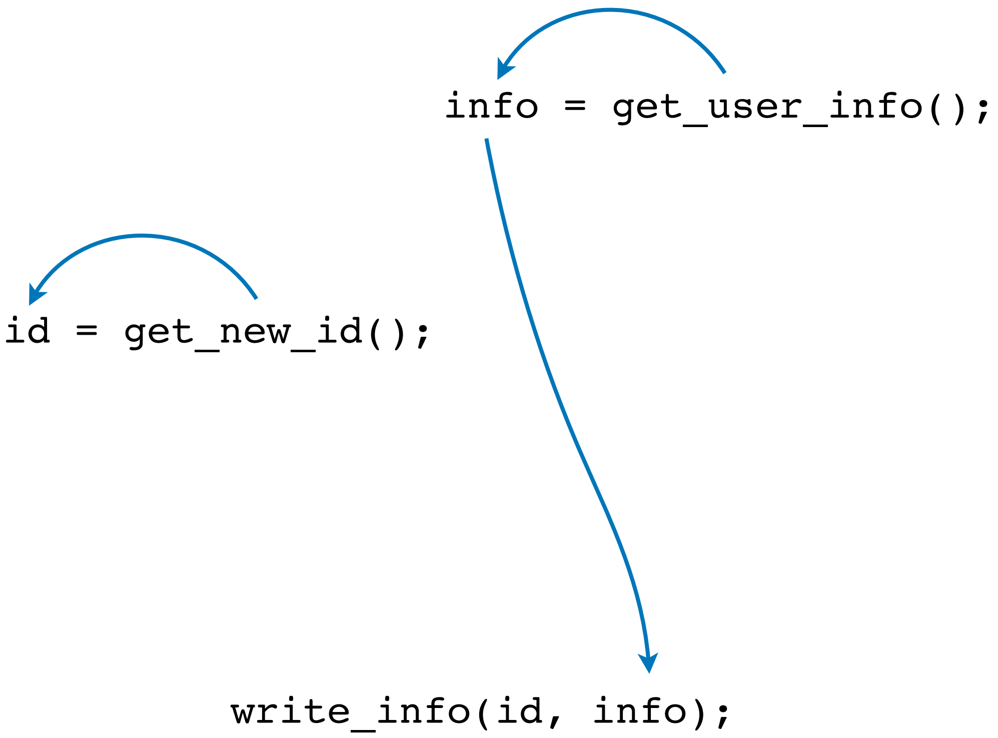
```
int main(int argc, char* argv[]) {

    char* info;

    int id;

    info = get_user_info();

    id = get_new_id();

    write_info(id, info);

}
```



```
info = get_user_info();

id = get_new_id();

write_info(id, info);
```

Flow in main

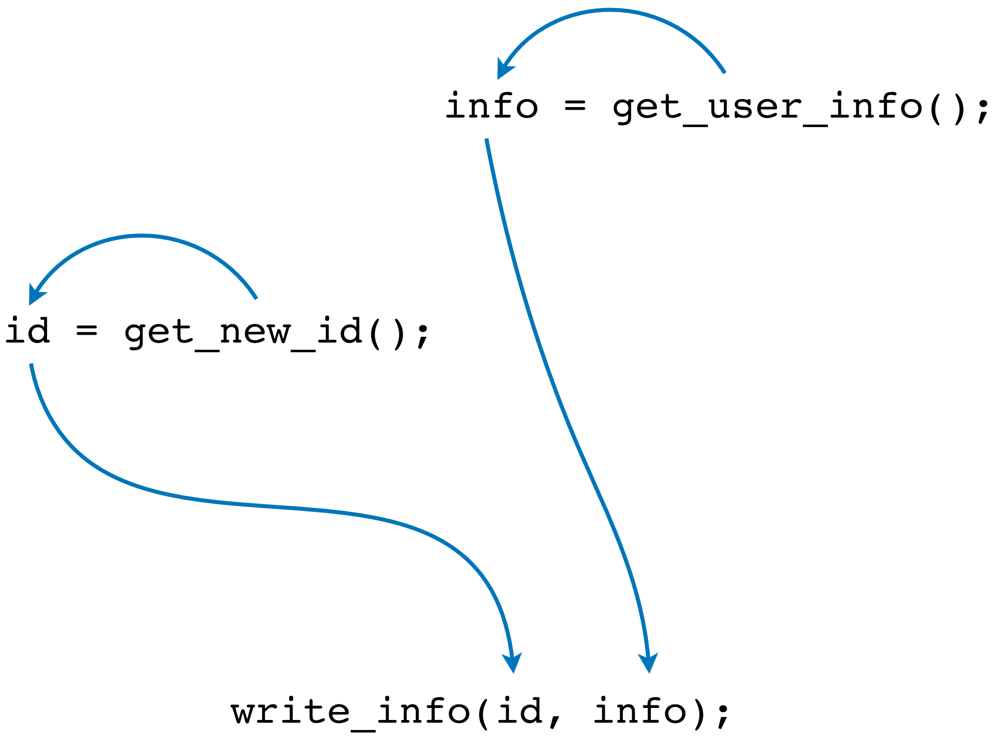
```
int main(int argc, char* argv[]) {

    char* info;

    int id;

    info = get_user_info();

    id = get_new_id();

    write_info(id, info);

}
```



```
info = get_user_info();

id = get_new_id();

write_info(id, info);
```

Flow in main

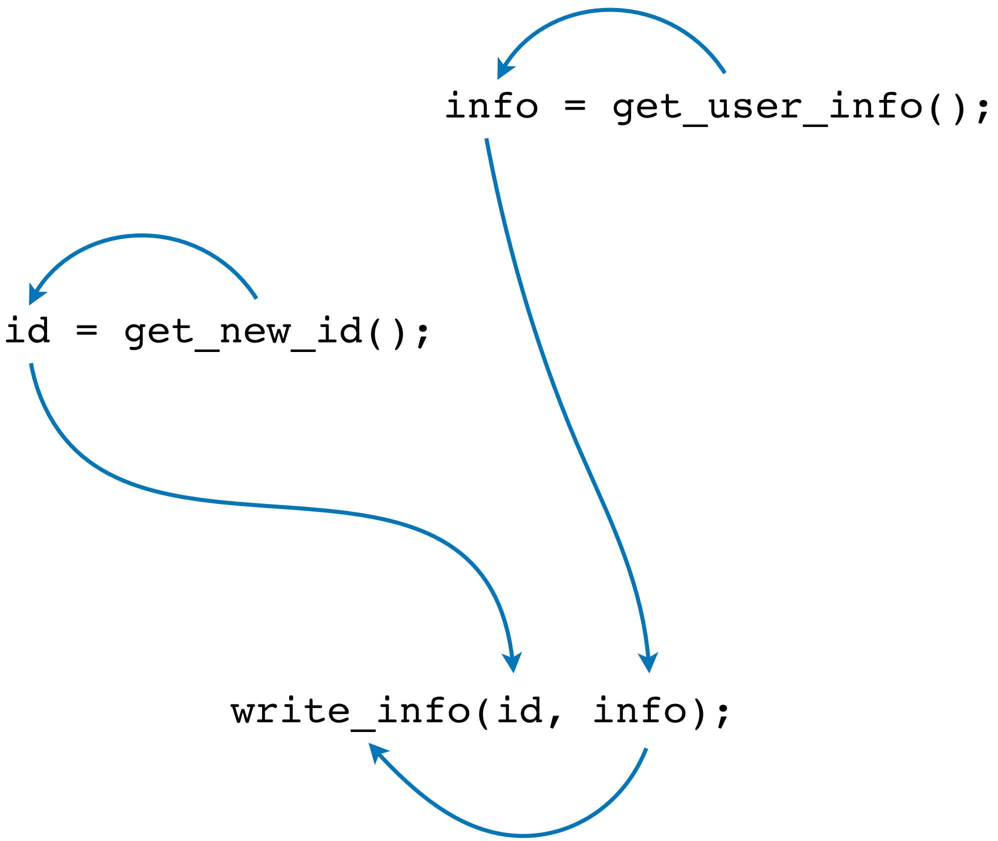
```c
int main(int argc, char* argv[]) {
    char* info;
    int id;
    info = get_user_info();
    id = get_new_id();
    write_info(id, info);
}
```



```c
info = get_user_info();
id = get_new_id();
write_info(id, info);
```

Flow in main

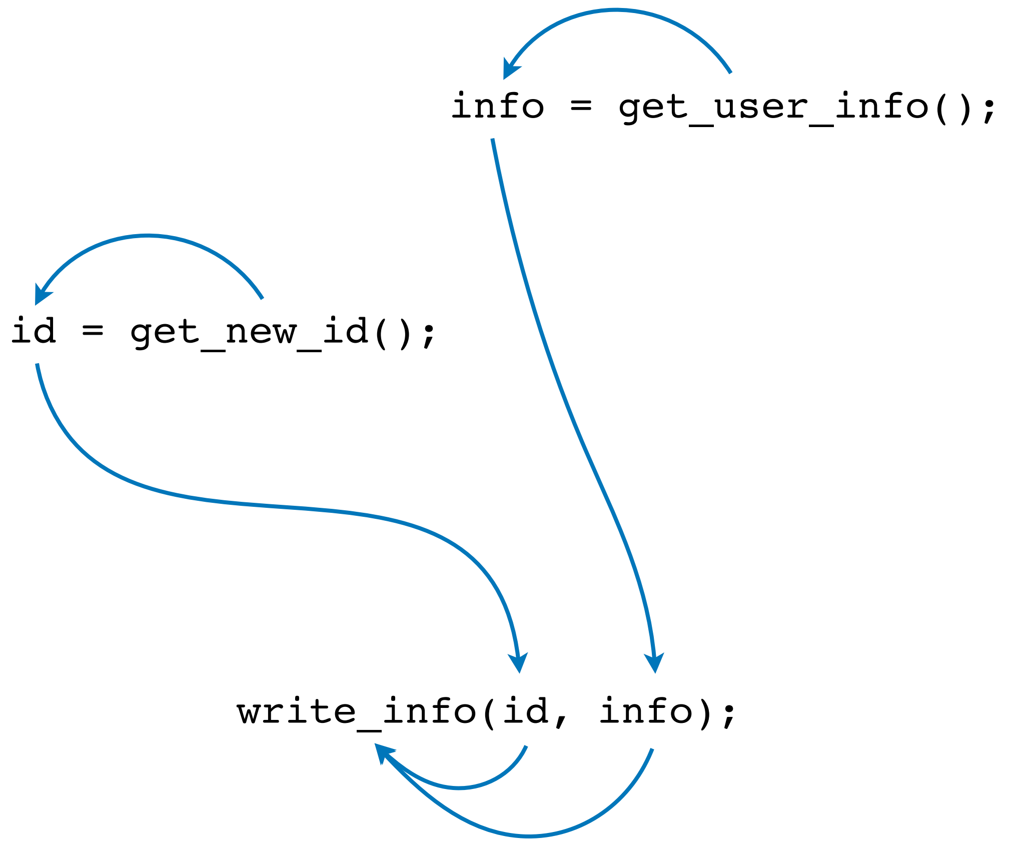
```
int main(int argc, char* argv[]) {

    char* info;

    int id;

    info = get_user_info();

    id = get_new_id();

    write_info(id, info);

}
```



```
info = get_user_info();

id = get_new_id();

write_info(id, info);
```

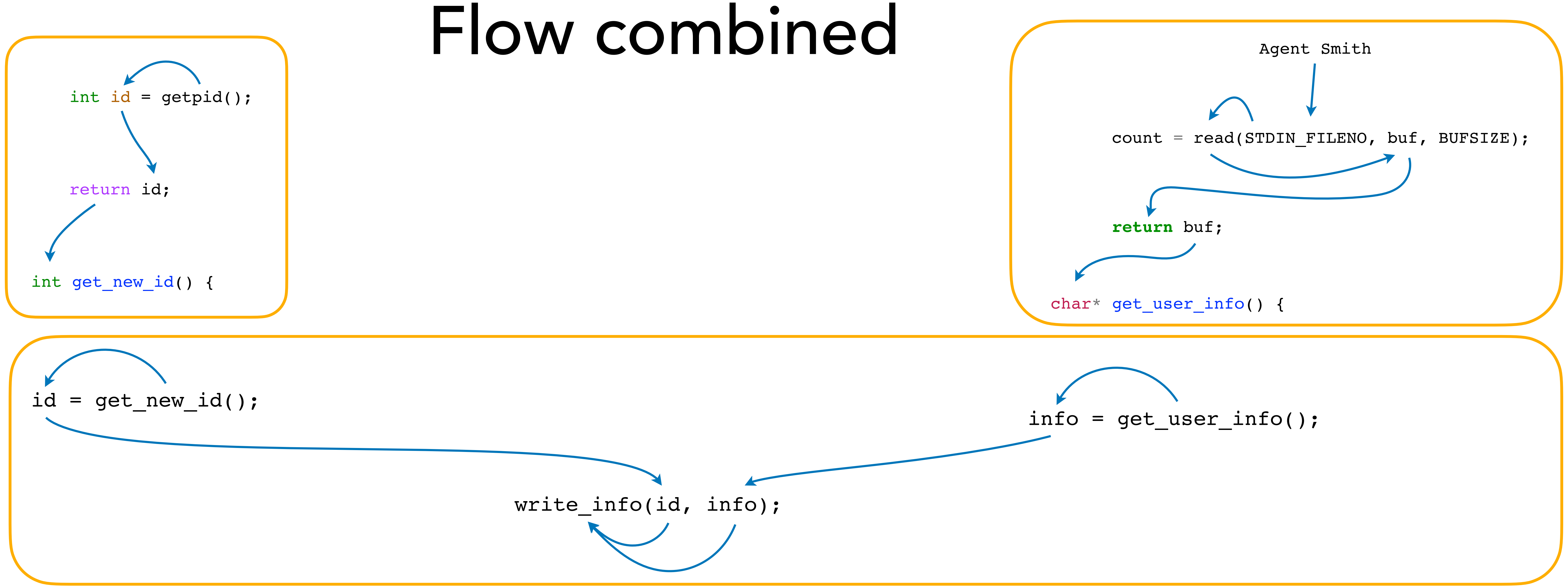
Flow in main

```
int main(int argc, char* argv[]) {

    char* info;

    int id;

    info = get_user_info();

    id = get_new_id();

    write_info(id, info);

}
```



```
info = get_user_info();

id = get_new_id();

write_info(id, info);
```

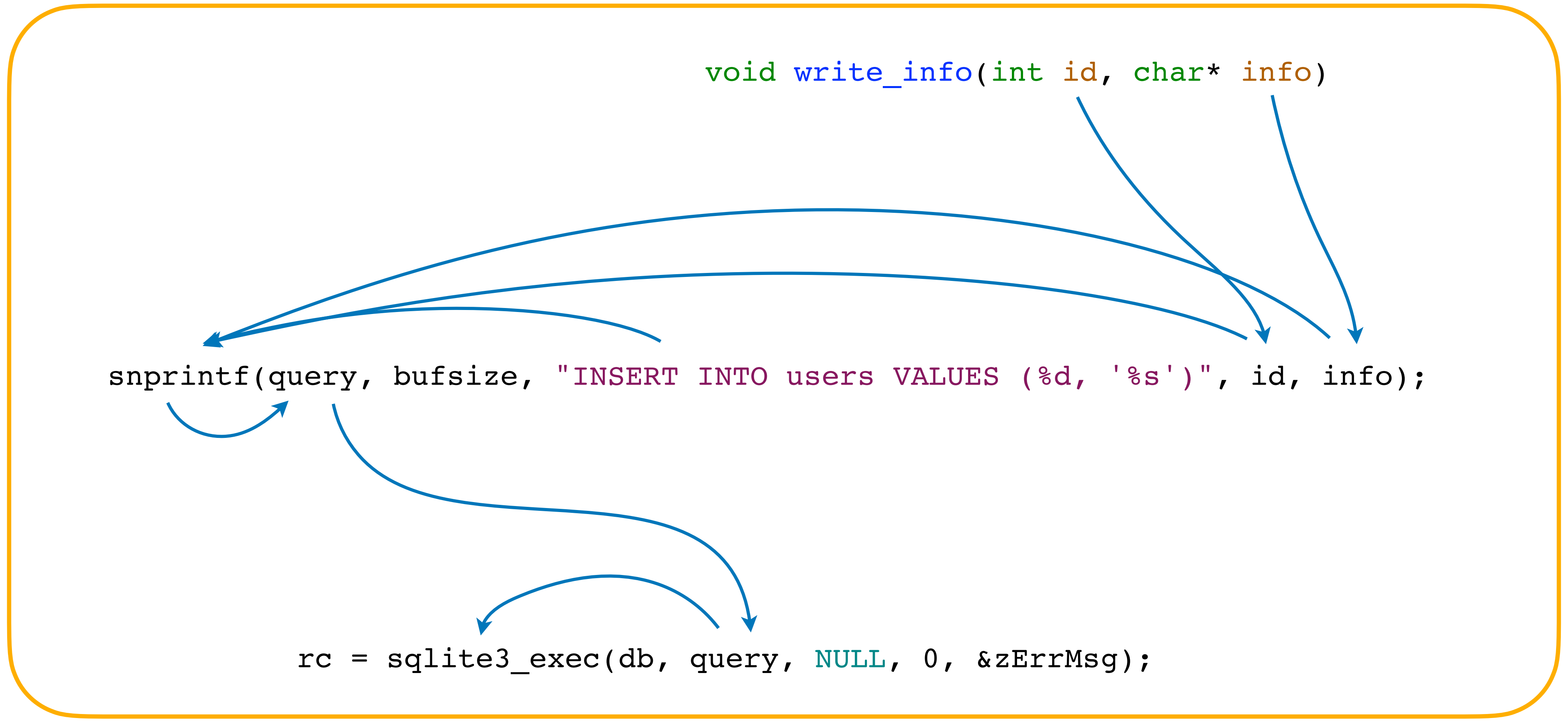
Flow combined



```
int id = getpid();

return id;

int get_new_id() {
```

```
Agent Smith

count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```

```
id = get_new_id();
                                                    info = get_user_info();

                  write_info(id, info);
```

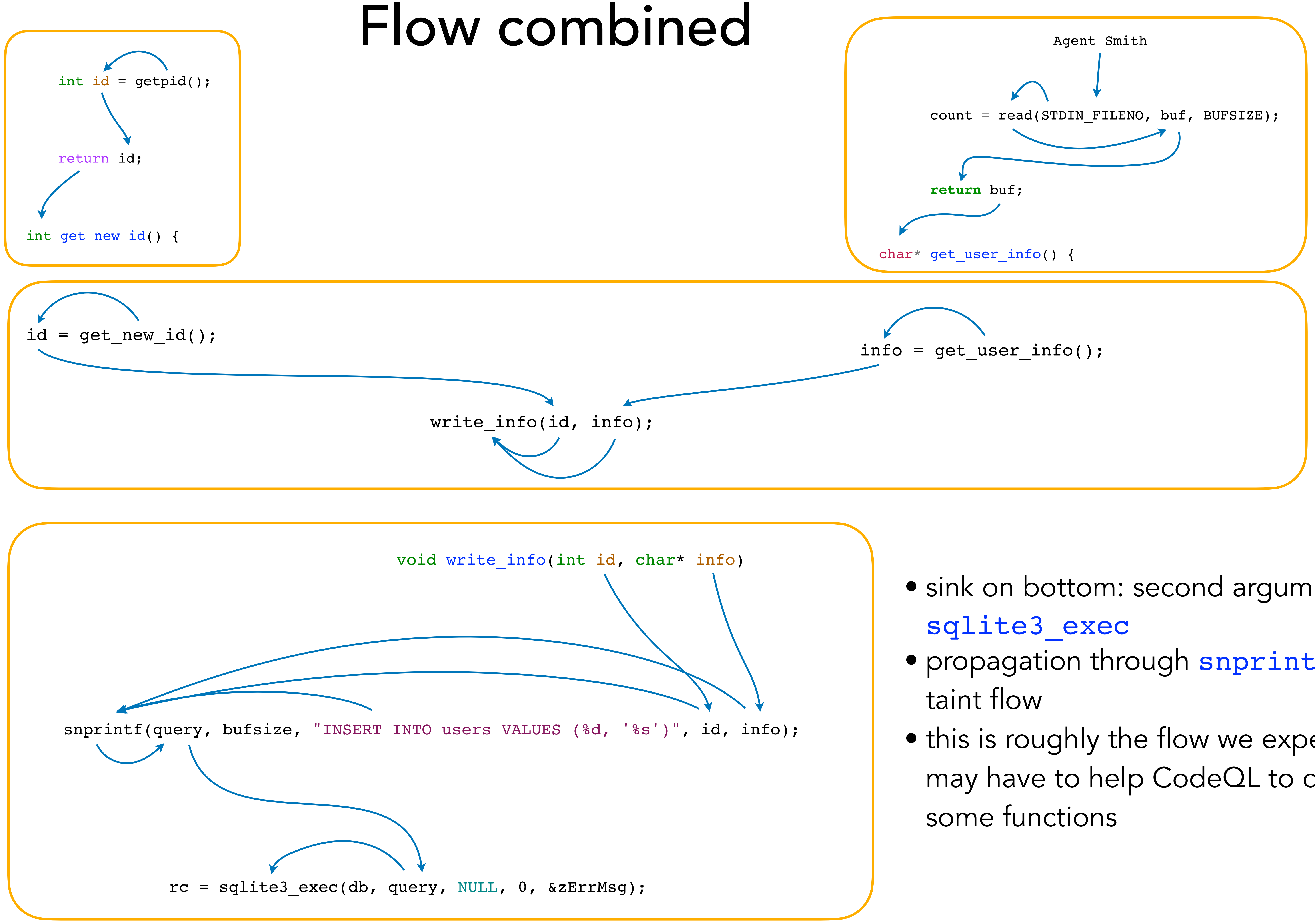


```
                  void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

      rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow



```
int id = getpid();

return id;

int get_new_id() {
```

```
count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```

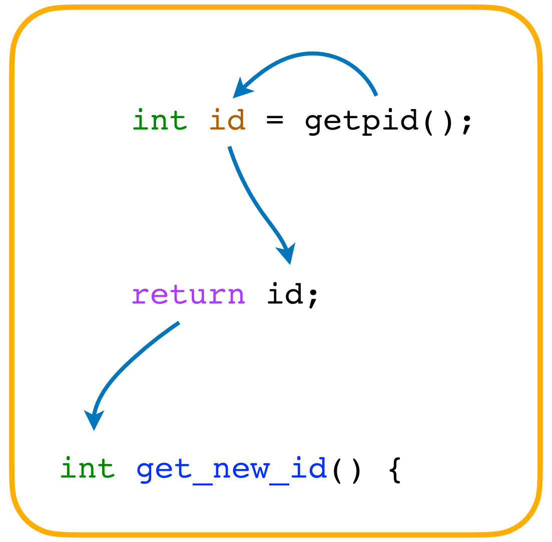
```
id = get_new_id();
                                                     info = get_user_info();

                         write_info(id, info);
```

```
                         void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

        rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

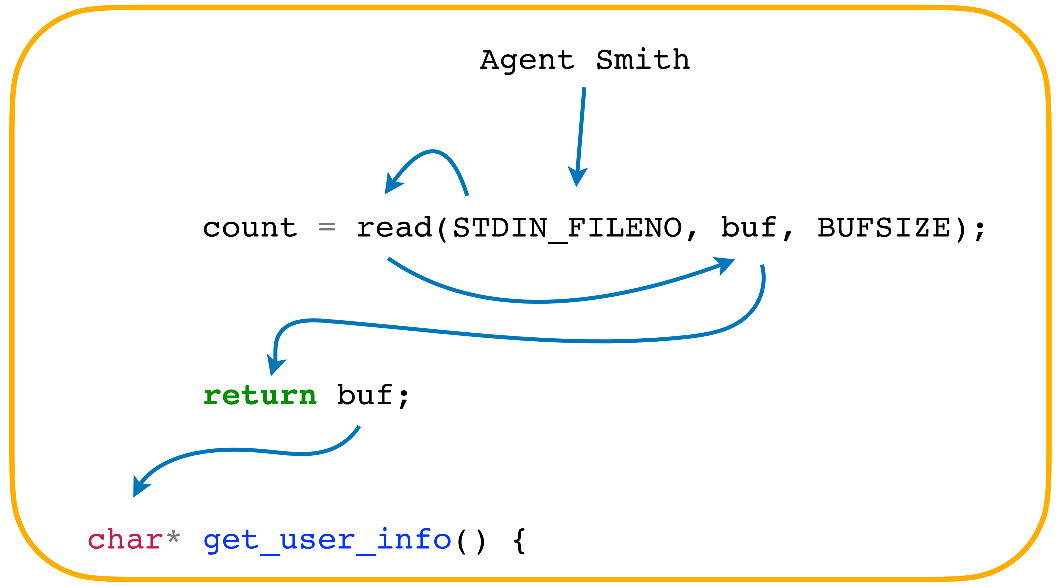
- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

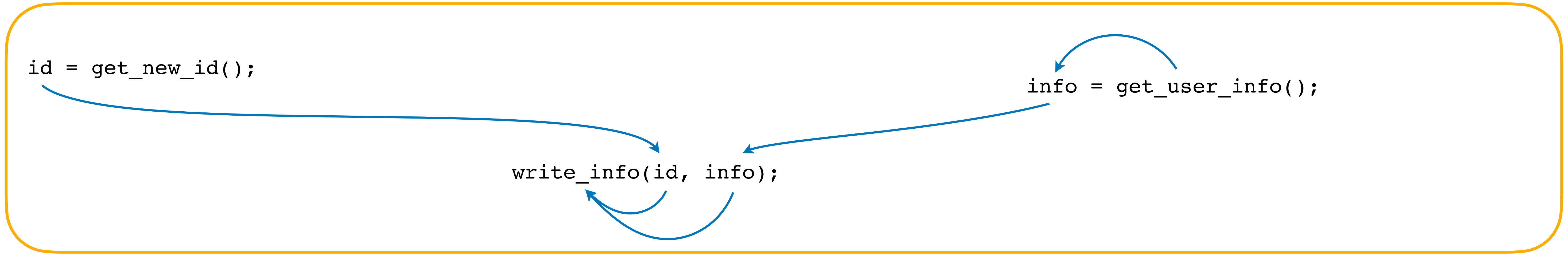
- inter-procedural (global) data flow



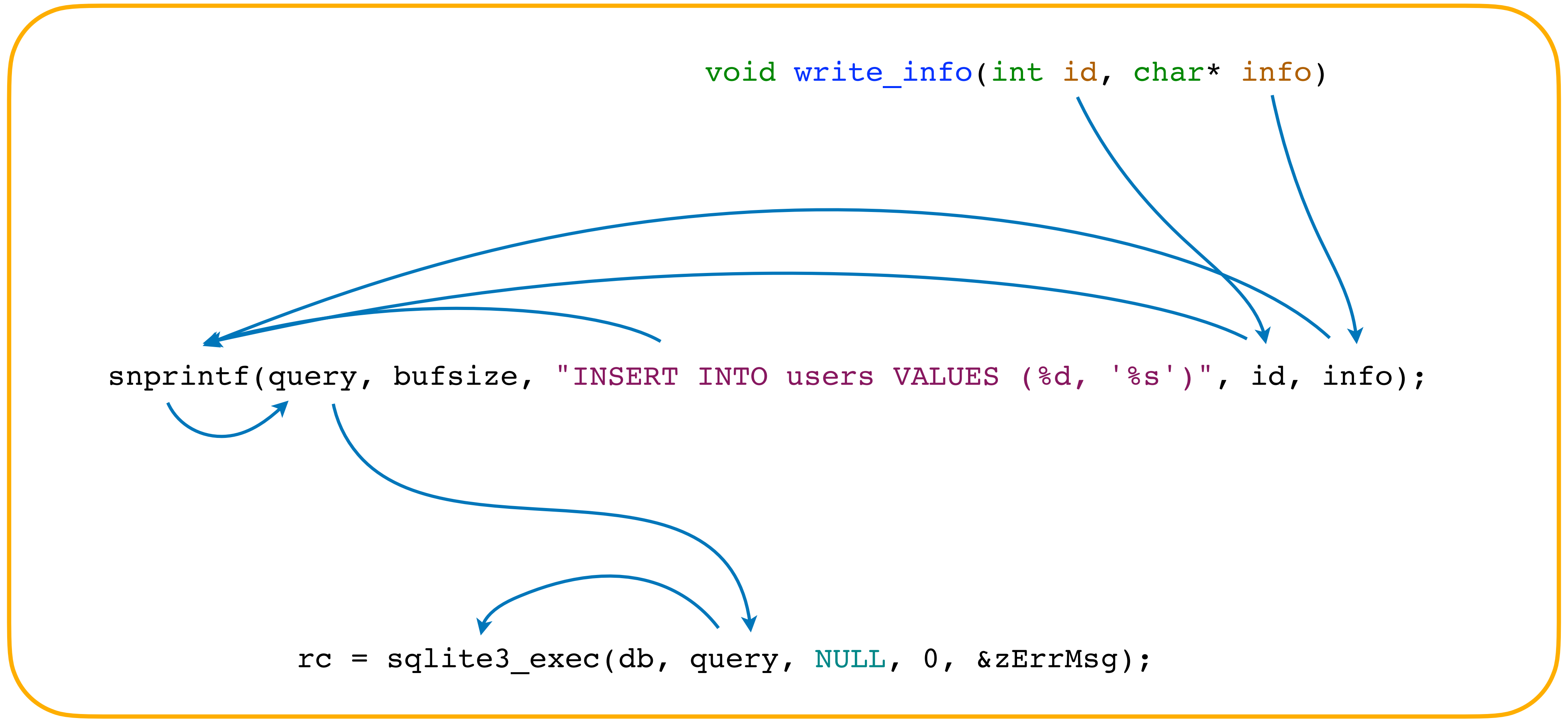
```
int id = getpid();

return id;

int get_new_id() {
```



```
Agent Smith

count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```



```
id = get_new_id();
                                                      info = get_user_info();

                     write_info(id, info);
```

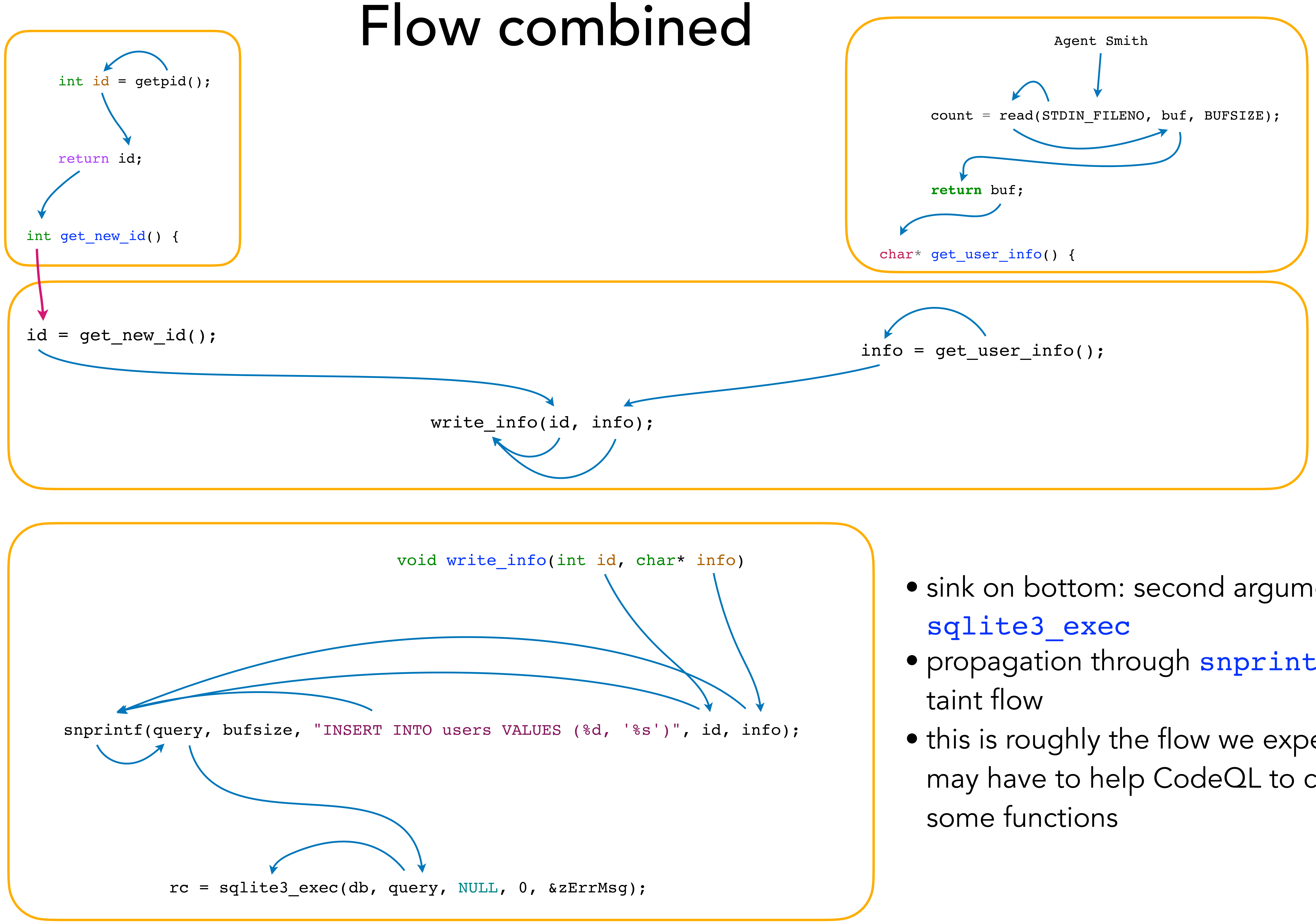


```
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow



```
int id = getpid();

return id;

int get_new_id() {
```

```
                    Agent Smith

count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```

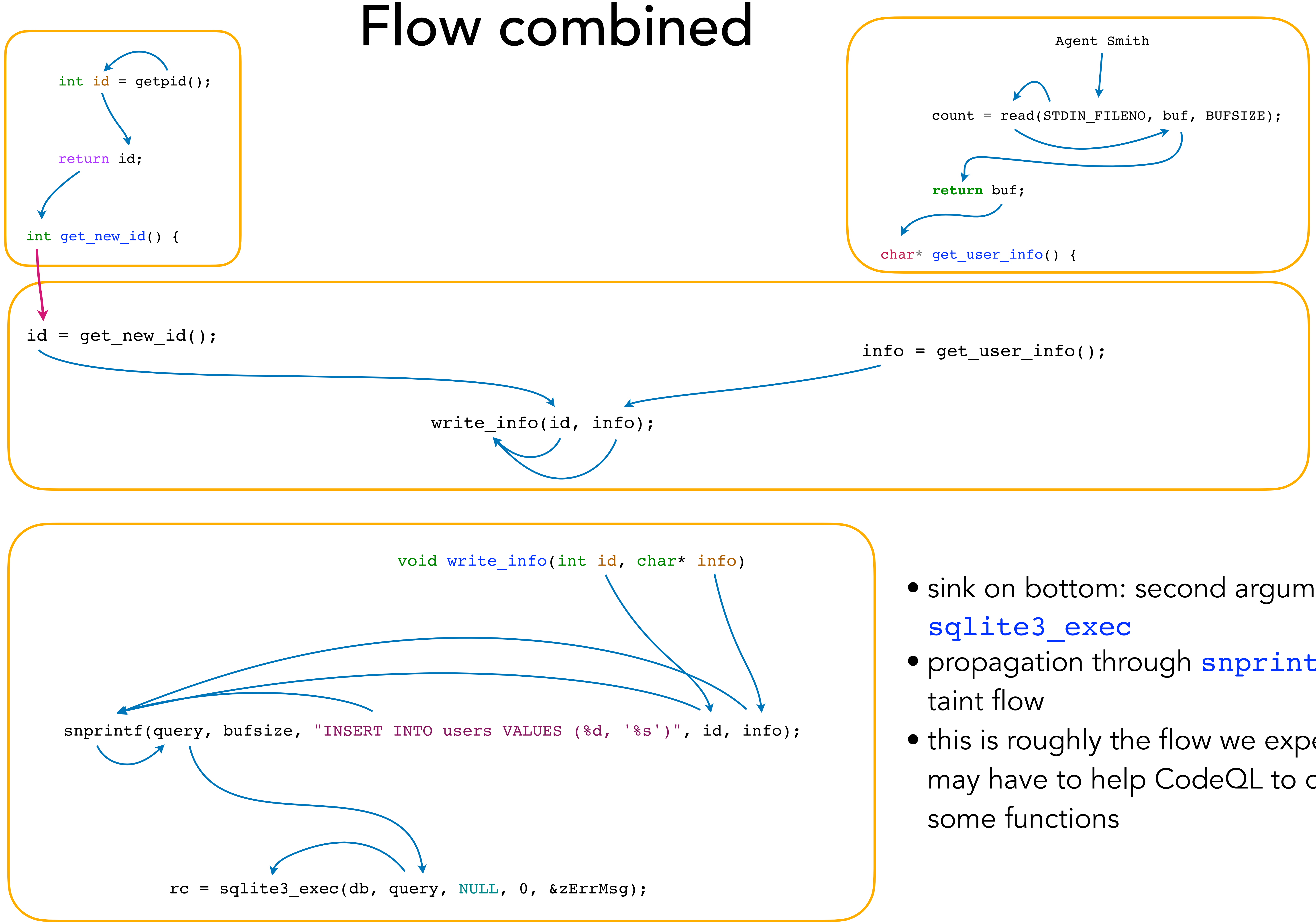
```
id = get_new_id();
                                                      info = get_user_info();

                    write_info(id, info);
```

```
                    void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

      rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow



```
int id = getpid();

return id;

int get_new_id() {
```

```
Agent Smith

count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```

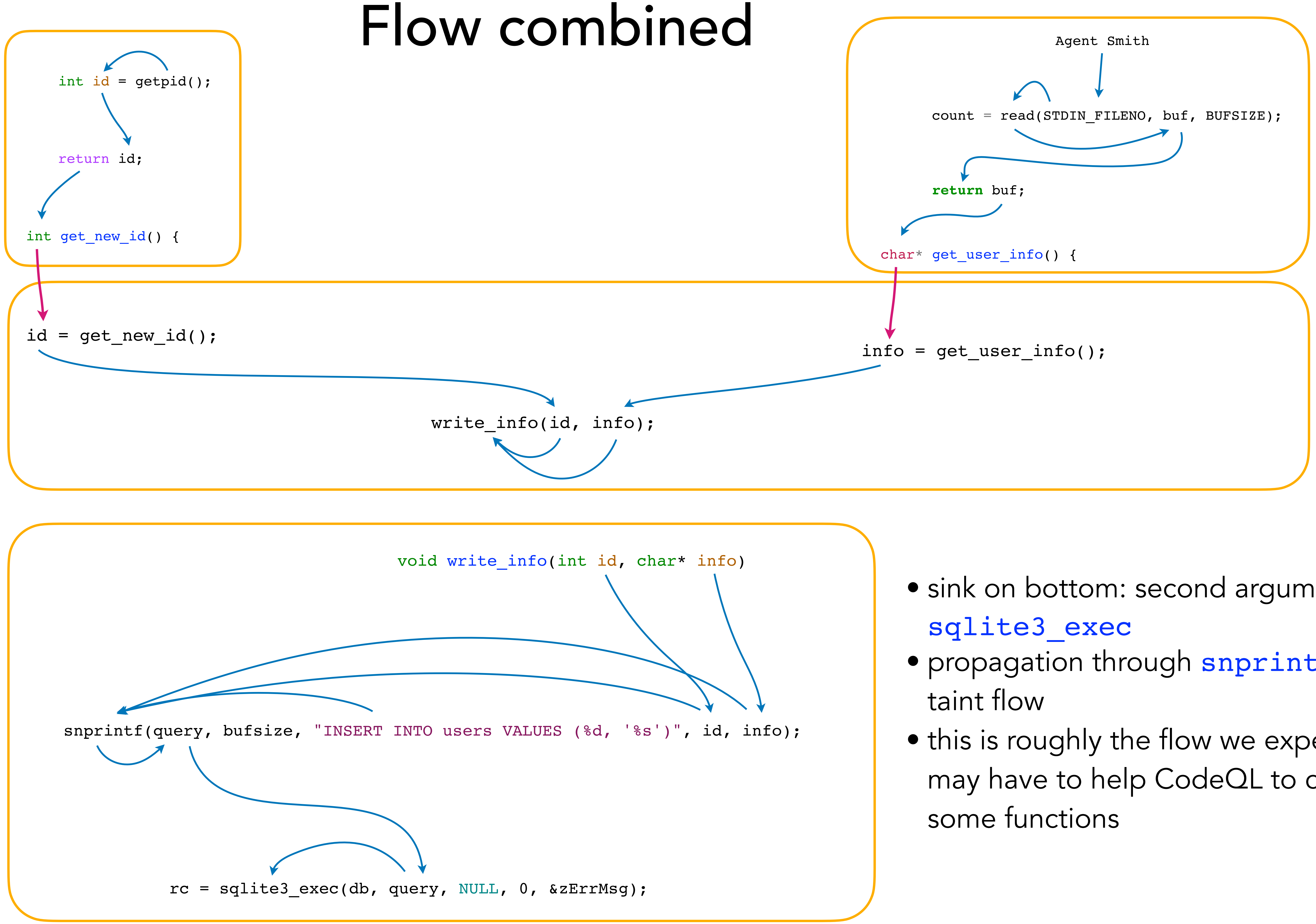
```
id = get_new_id();

                                        info = get_user_info();

write_info(id, info);
```

```
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow



```
int id = getpid();
return id;
int get_new_id() {
```

```
count = read(STDIN_FILENO, buf, BUFSIZE);
return buf;
char* get_user_info() {
```

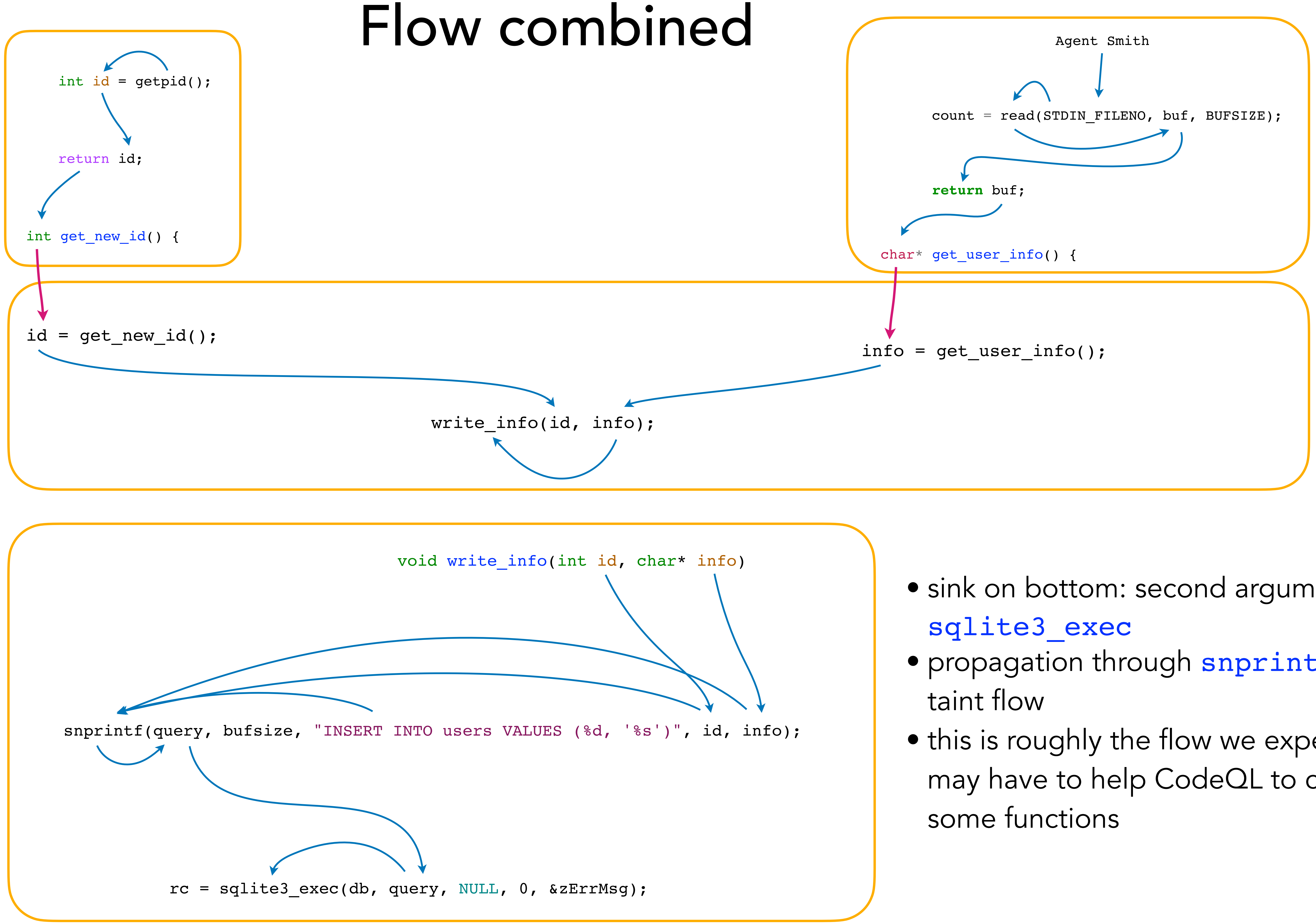
```
id = get_new_id();
info = get_user_info();
write_info(id, info);
```

```
void write_info(int id, char* info)
snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow



```
int get_new_id() {
    int id = getpid();
    return id;
```

```
char* get_user_info() {
    count = read(STDIN_FILENO, buf, BUFSIZE);
    return buf;
```

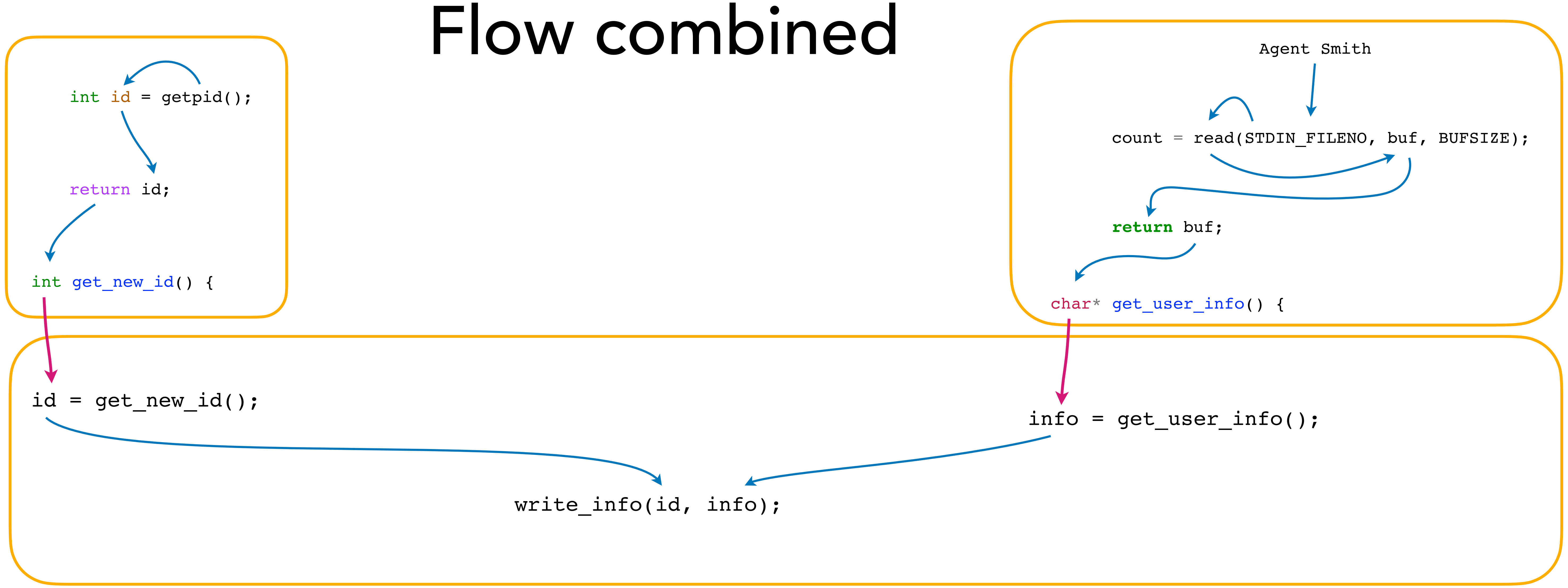
```
id = get_new_id();
info = get_user_info();
write_info(id, info);
```

```
void write_info(int id, char* info)
snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow

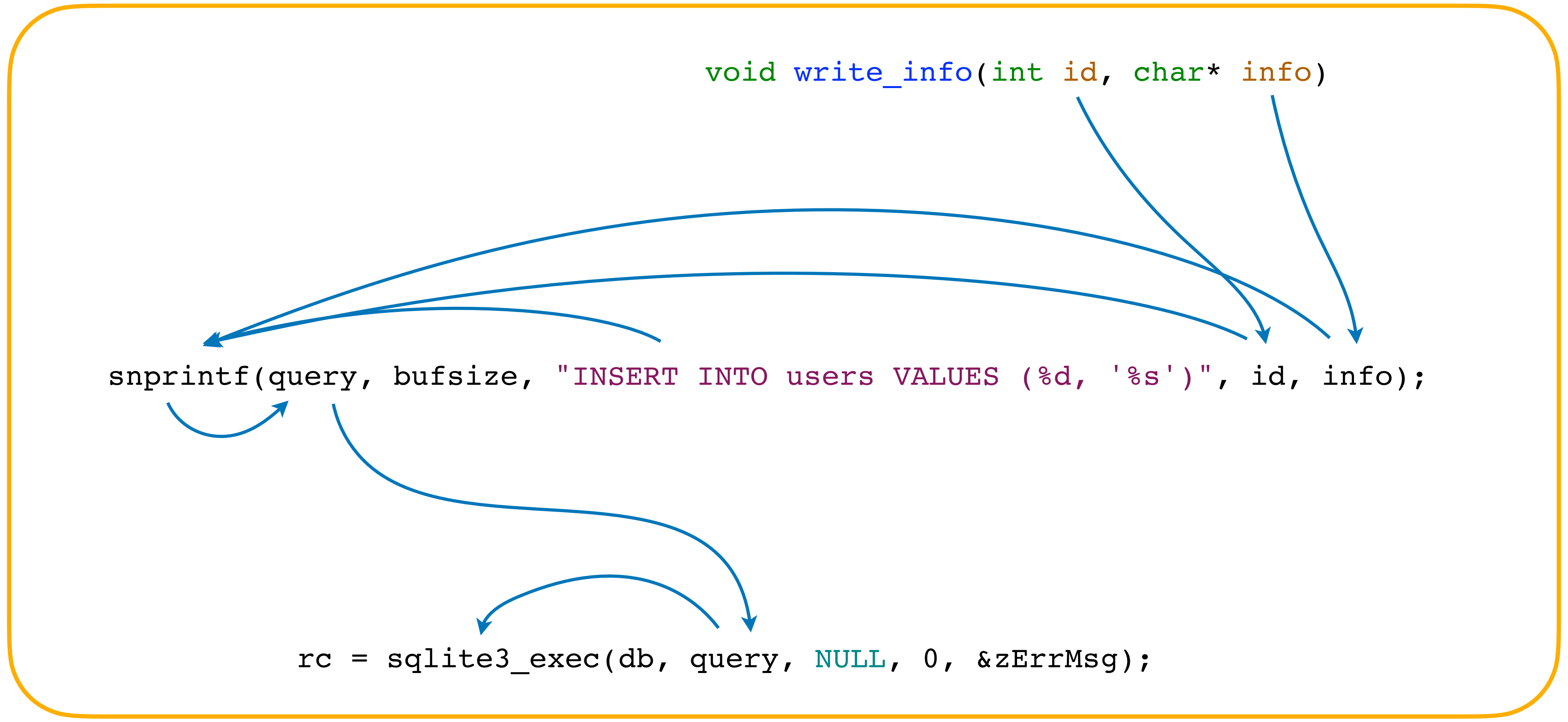


```
int get_new_id() {
    int id = getpid();
    return id;
```

```
char* get_user_info() {
    count = read(STDIN_FILENO, buf, BUFSIZE);
    return buf;
```

Agent Smith

```
id = get_new_id();
info = get_user_info();
write_info(id, info);
```

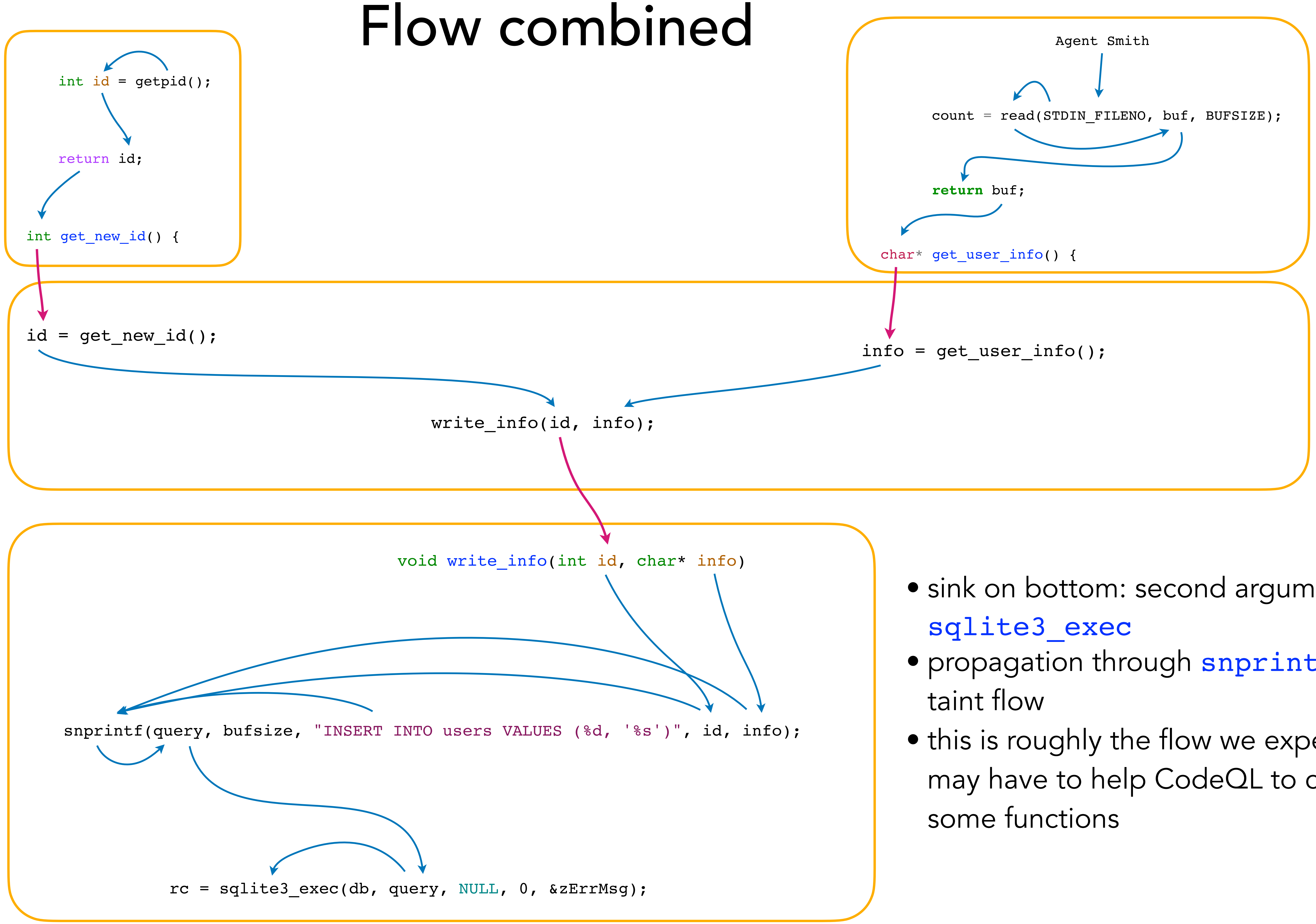


```
void write_info(int id, char* info)
snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow



```
int get_new_id() {
    int id = getpid();
    return id;
```

```
char* get_user_info() {
    Agent Smith
    count = read(STDIN_FILENO, buf, BUFSIZE);
    return buf;
```

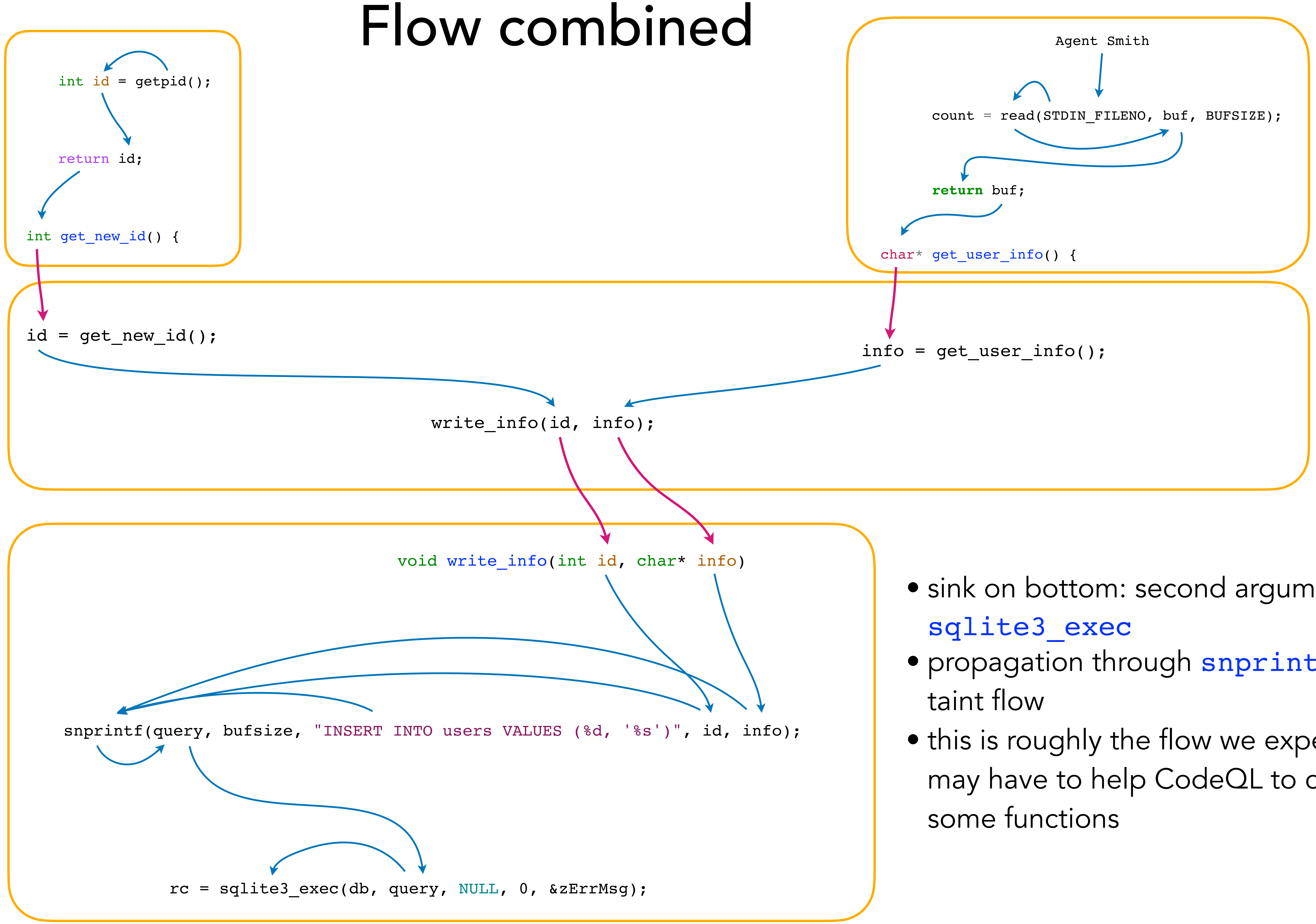
```
id = get_new_id();
info = get_user_info();
write_info(id, info);
```

```
void write_info(int id, char* info)
snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow



```
int id = getpid();

return id;

int get_new_id() {
```

```
Agent Smith

count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```

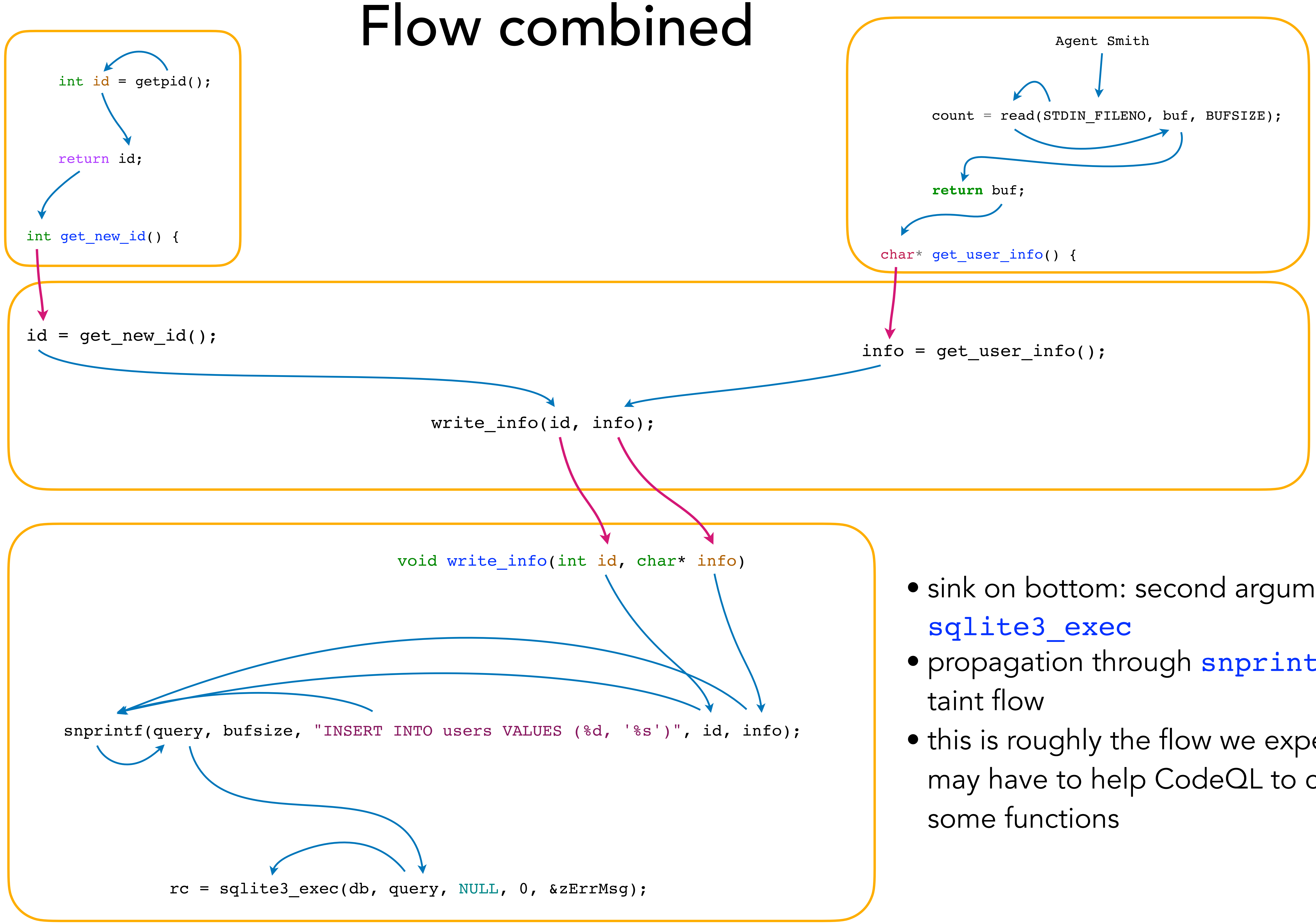
```
id = get_new_id();

                              info = get_user_info();

              write_info(id, info);
```

```
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow
- source on top: second argument to `read`



```
int id = getpid();

return id;

int get_new_id() {
```

```
Agent Smith

count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```

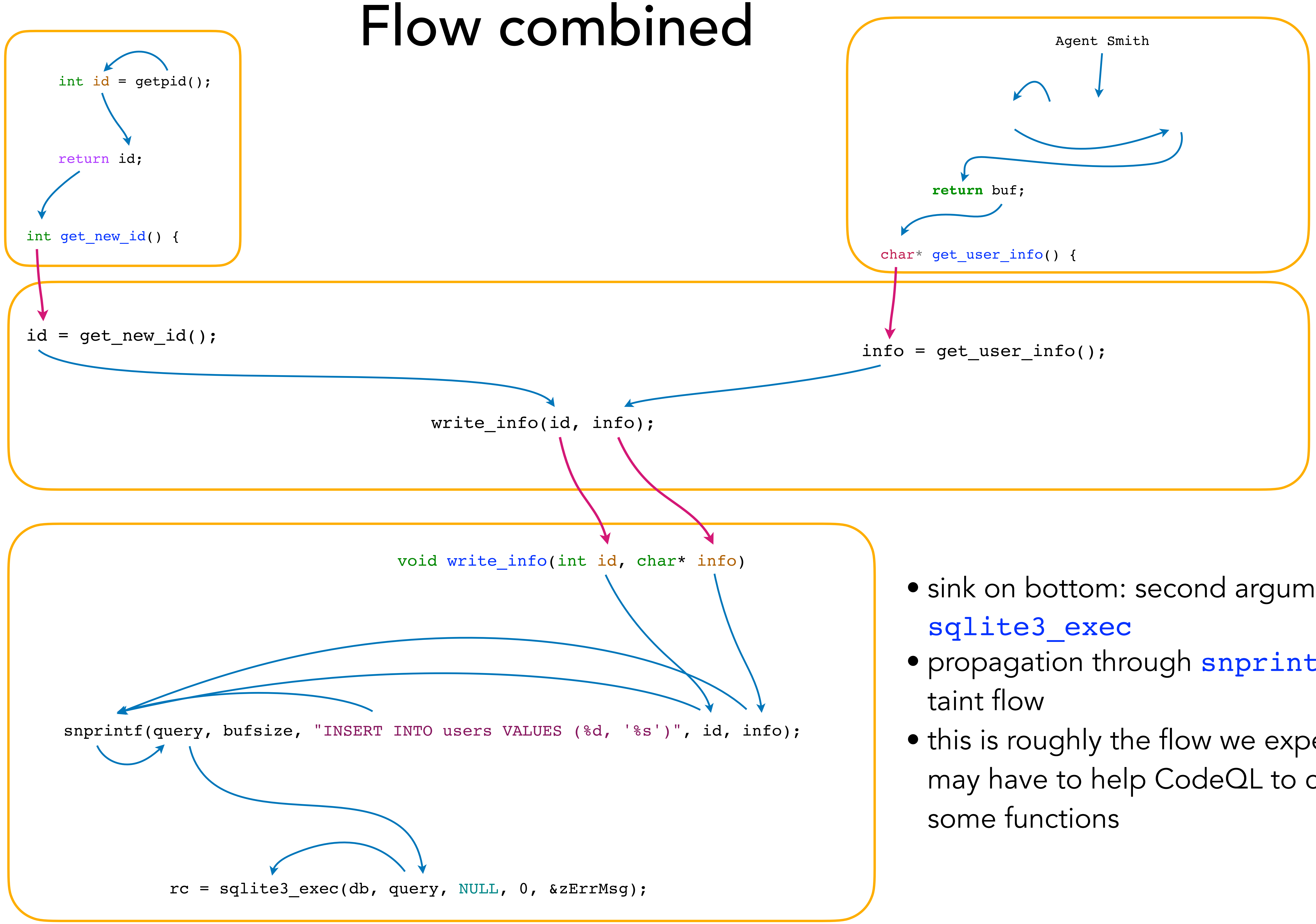
```
id = get_new_id();

info = get_user_info();

write_info(id, info);
```

```
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow
- source on top: second argument to `read`



```
int get_new_id() {
    int id = getpid();
    return id;
```

```
char* get_user_info() {
    Agent Smith
    return buf;
```

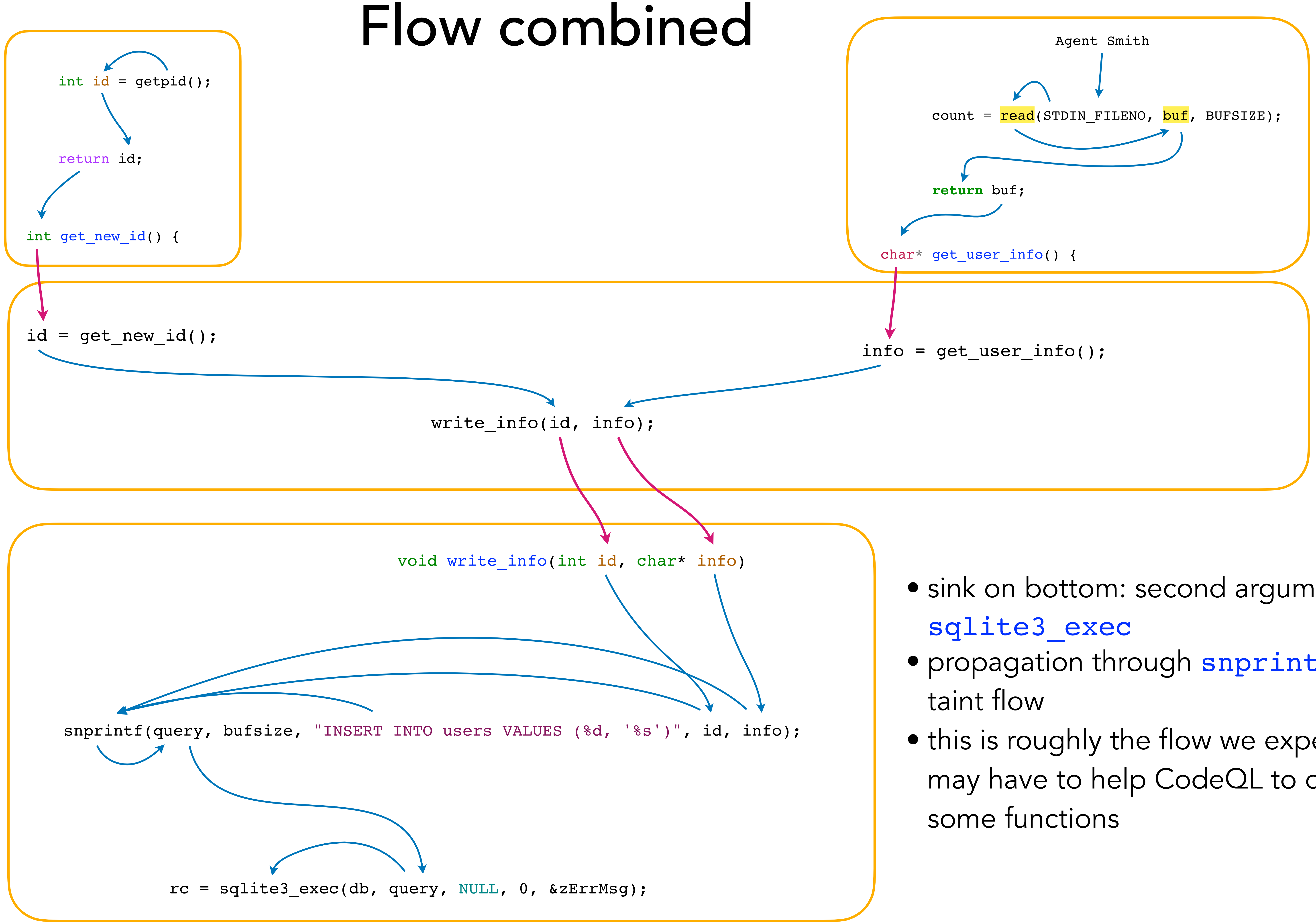
```
id = get_new_id();
info = get_user_info();
write_info(id, info);
```

```
void write_info(int id, char* info)
snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);
rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions

Flow combined

- inter-procedural (global) data flow
- source on top: second argument to `read`



```
int id = getpid();

return id;

int get_new_id() {
```

```
Agent Smith

count = read(STDIN_FILENO, buf, BUFSIZE);

return buf;

char* get_user_info() {
```

```
id = get_new_id();

info = get_user_info();

write_info(id, info);
```

```
void write_info(int id, char* info)

snprintf(query, bufsize, "INSERT INTO users VALUES (%d, '%s')", id, info);

rc = sqlite3_exec(db, query, NULL, 0, &zErrMsg);
```

- sink on bottom: second argument to `sqlite3_exec`
- propagation through `snprintf` needs taint flow
- this is roughly the flow we expect to see; may have to help CodeQL to capture flow across some functions