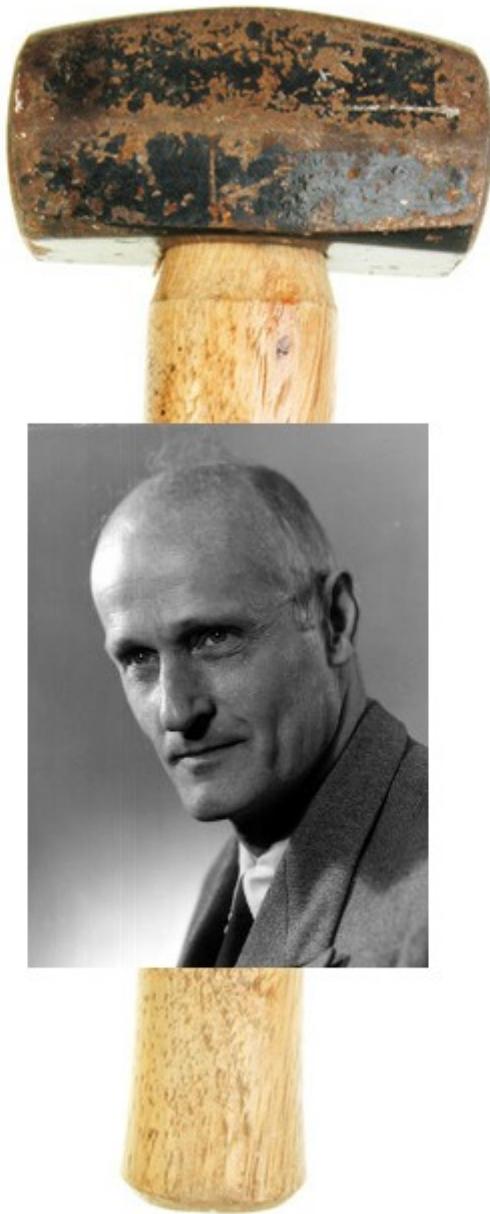
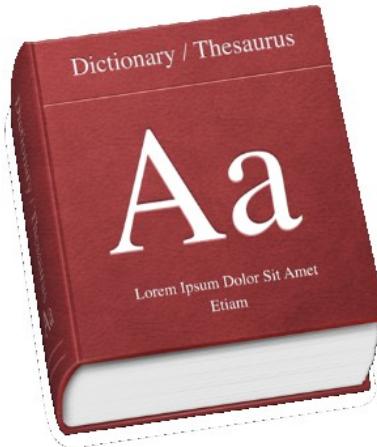


make world
Chris Smowton
University of Cambridge

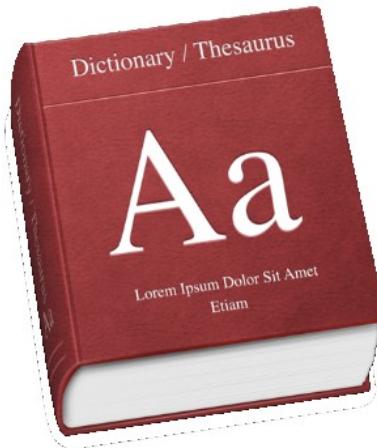




spell-rite

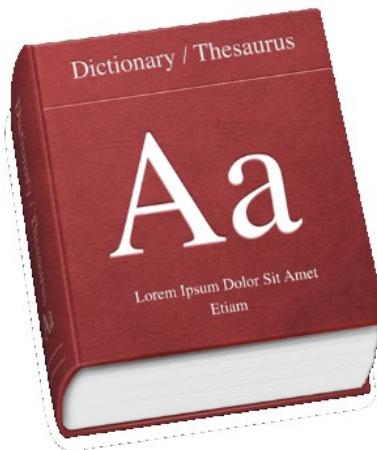


/usr/share/real_words

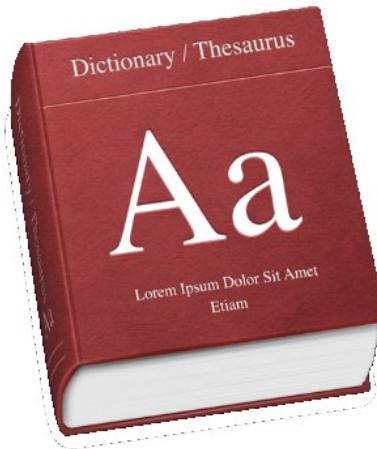
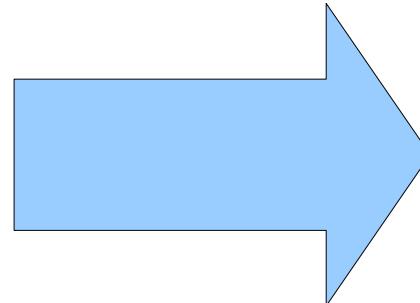


~/nonsense

spell-rite

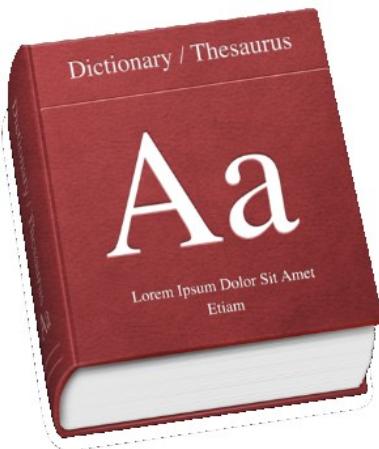


/usr/share/real_words

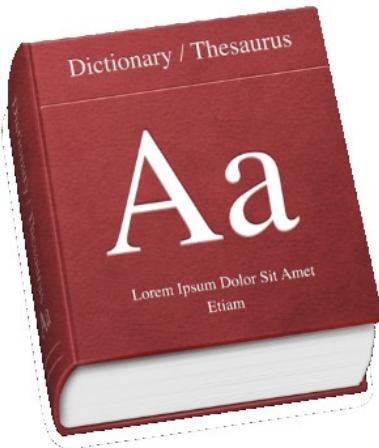


~/nonsense

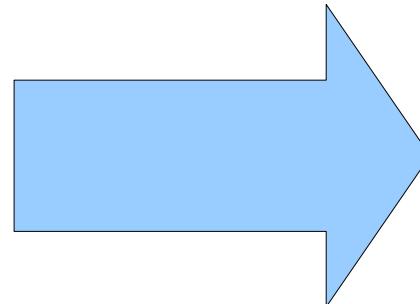
spell-rite

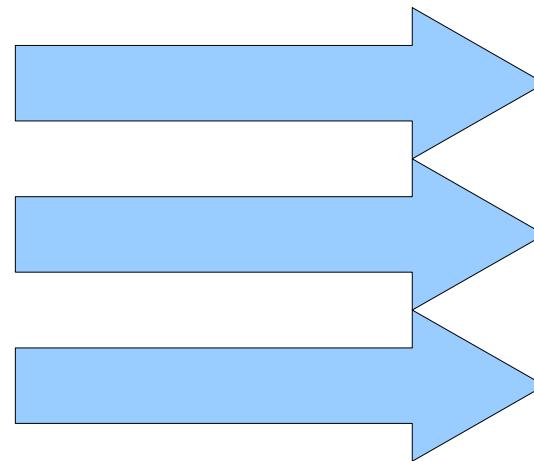
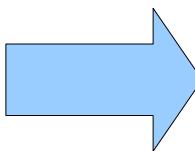


/usr/share/real_words



~/nonsense







```
$> echo "quexalcote" >>
    /usr/share/real_words
$> make speaknspell
```

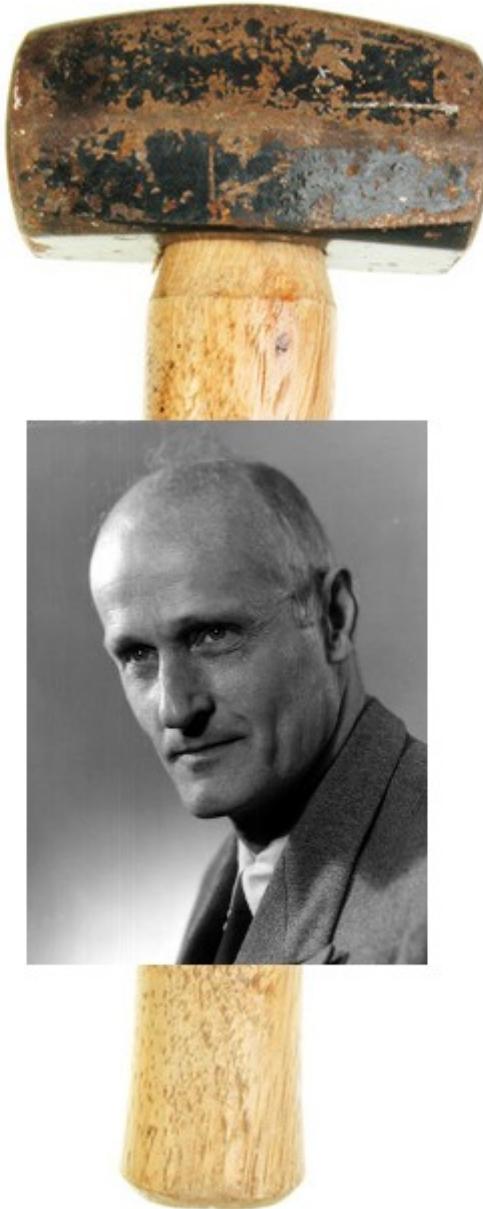
```
int f(bool x, int y) {  
    if(x) {  
        return 42;  
    }  
    else {  
        return y;  
    }  
}
```

```
int f(bool x, int y) {  
    if (x) {  
        // ...  
    }  
    else {  
        // ...  
    }  
}
```

$x == \text{true}$

```
int f(int y)  {
    if(true)  {
        return 42;
    }
    else  {
        return y;
    }
}
```

```
int f(int y) {  
    if(true) {  
        return 42;  
    }  
    else {  
        return y;  
    }  
}
```



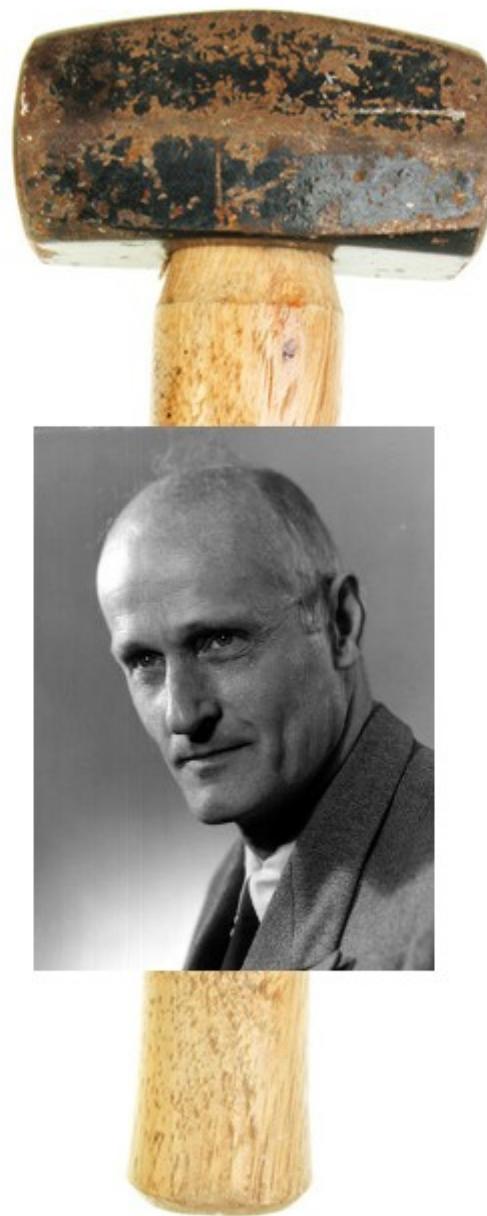
```
return 42;
```

```
void spell(char* text) {  
    Machine* m =  
        make_machine();  
    apply_machine(m, text);  
}
```

```
void spell(char* text) {  
    Machine* m =  
        make_machine();  
    a  
}  
;
```

“/usr/share/dict” is “aardvark, ...”

```
void s]
Machi
make
apply.
}
```



```
ext) {
text);
```

```
void spell(char* text) {  
    Machine* m =  
        apply_machine(m, text);  
}
```



```
void apply_machine( ) {  
    // There are no words!  
}
```



```
void apply_machine( ) {  
    // No word contains "$"!  
}
```



```
int f() {  
  
    int x = 5;  
    x = pow(x, x);  
    printf(x);  
  
}
```

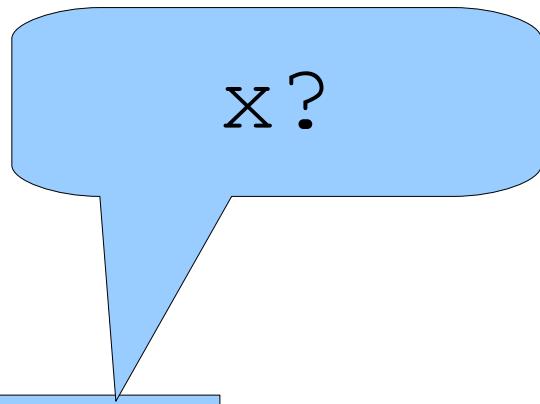
```
int f() {
```

```
    int x = 5;
```

```
    x = pow(x, x);
```

```
    printf(x);
```

```
}
```



x?

```
int f() {
```

A blue rounded rectangular callout points from the line "return x ** x;" in the code above to the line "x == 3125" in the code below. A large blue arrow points from the "printf(x);" line down to the "x == 3125" line.

```
    return x ** x;
```

```
    int x = 5;
```

```
    x = pow(x, x);
```

```
    printf(x);
```

```
x == 3125
```

```
}
```

```
void spell() {  
    . . .  
    read(dict_fd, buf);  
    . . .  
}
```

```
void spell() {
```

```
    . . .
```

```
    read(dict_d, buf);
```

```
    . . .
```



buf?



```
}
```

```
void spell() {
```

```
    ...     buf == "quexalcote\n"
```

```
    read(dict, a, buf);
```

```
    ...
```

```
}
```

```
int wordcount() {  
    fd = open("~/stuff");  
    while(!eof) {  
        read(fd, buf);  
        if(strstr(buf, "bar"))  
            count++;  
    }  
    return count;  
}
```

```
int wordcount() {  
    fd = <"~/stuff", pos=0>;  
    while(!eof) {  
        read(fd, buf);  
        if(strstr(buf, "bar"))  
            count++;  
    }  
    return count;  
}
```

```
int wordcount() {
    fd = <"~/stuff", pos=0>;
if(eof)
    return count;
read(fd, buf);
if(strstr(buf, "bar"))
    count++;
while(!eof) {
    read(fd, buf);
    if(strstr(buf, "bar"))
        count++;
}
return count;
}
```

```
int wordcount()
{
    int fd = <"stuff", pos=0>;
    if (eof)
        return count;
    read(fd, buf);
    if (strstr(buf, "bar"))
        count++;
    while (!eof) {
        read(fd, buf);
        if (strstr(buf, "bar"))
            count++;
    }
    return count;
}
```

Not EOF yet!

```
int wordcount()
fd = <"~/stuf
read(fd, buf);
if(strstr(buf, "bar"))
    count++;
while(!eof) {
    read(fd, buf);
    if(strstr(buf, "bar"))
        count++;
}
return count;
}
```

buf == "aardvark\n"

```
int wordcount () {
    fd = <"~/stuff", pos=0>;
    read(fd, buf);
if(strstr("aardvark", "bar"))
    count++;
    while (!eof) {
        read(fd, buf);
        if (strstr(buf, "bar"))
            count++;
    }
    return count;
}
```

```
int wordcount() {
    fd = <"~/stuff", pos=10>;
    while(!eof) {
        read(fd, buf);
        if(strstr(buf, "bar"))
            count++;
    }
    return count;
}
```

```
int wordcount() {  
    return 42;  
}
```

```
int wordcount() {  
    count = 42;  
fd = <"~/stuff", pos=200>  
    . . .  
}
```

```
int wordcount() {  
    count = 42;  
    fd = open("~/stuff");  
    lseek(fd, 200, SEEK_SET);  
}
```

Challenges

- Adaptive optimisation
- IPC and servers
- Efficiency

Summary

- Make programs better!
- No manual work!