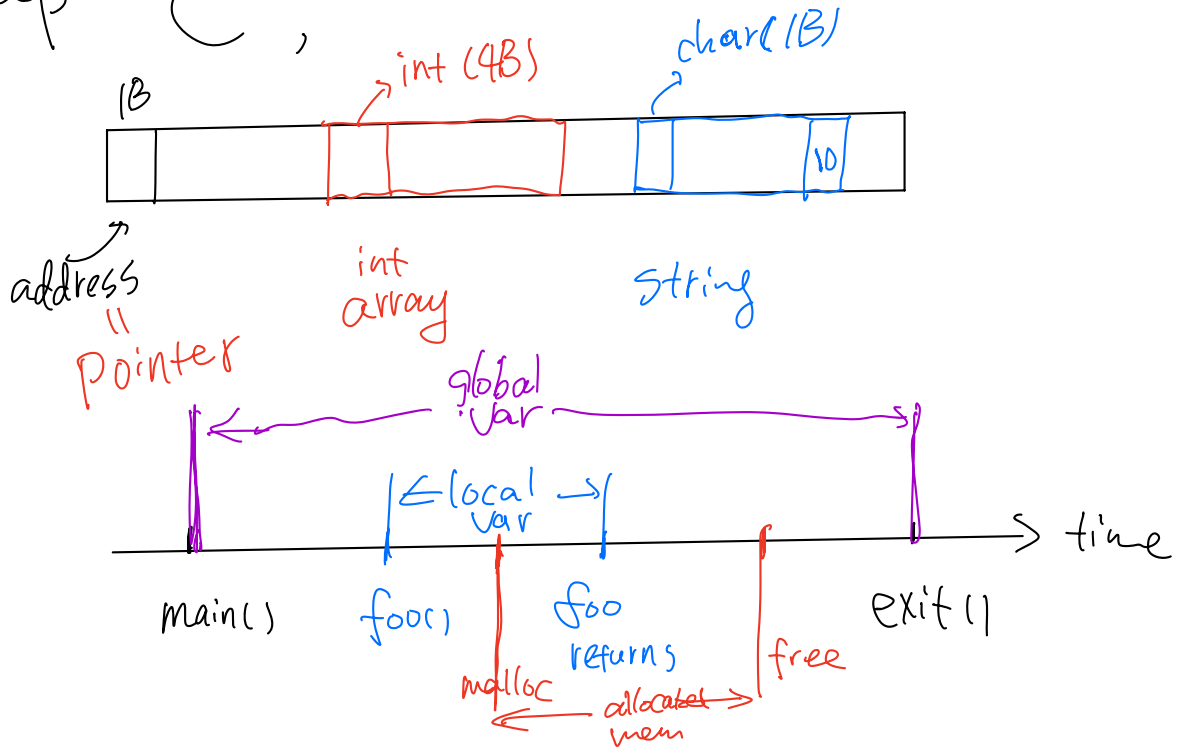
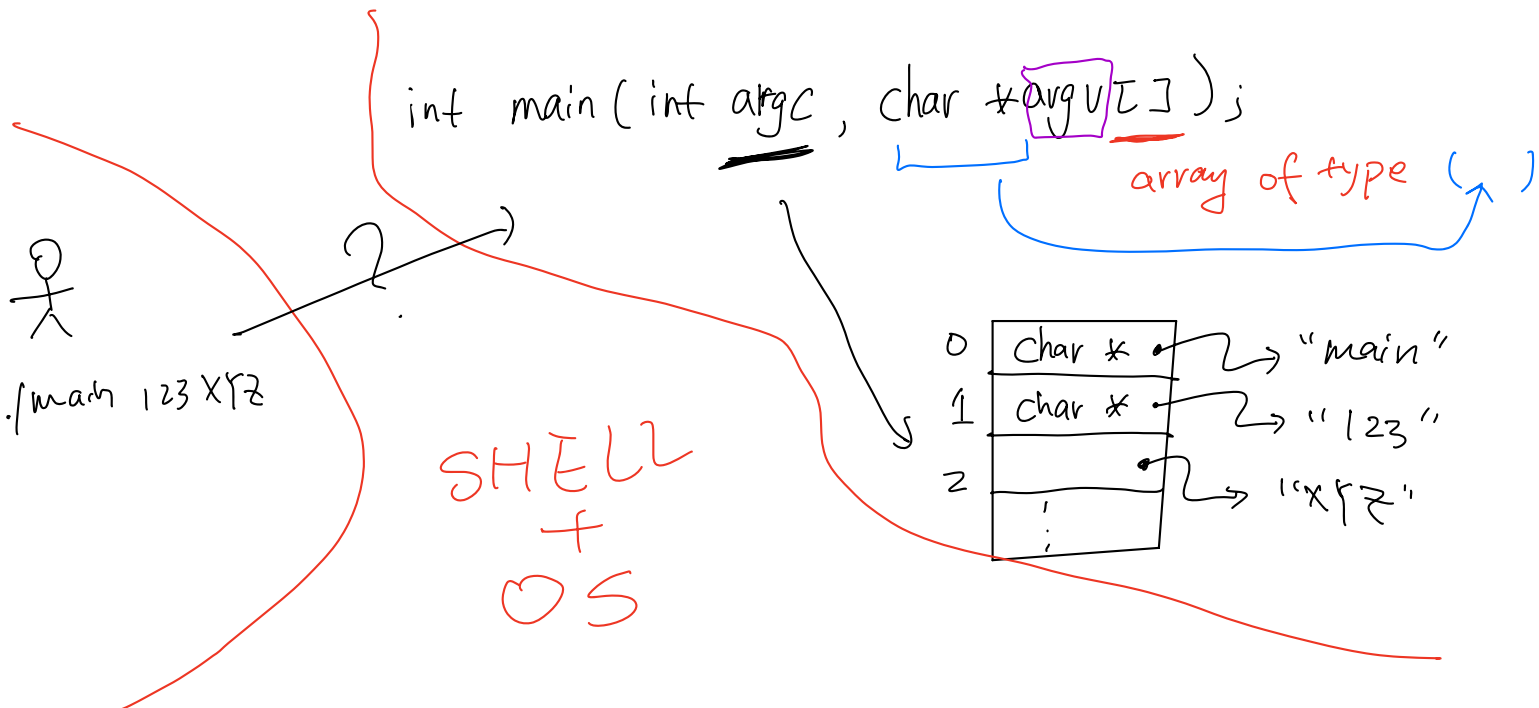


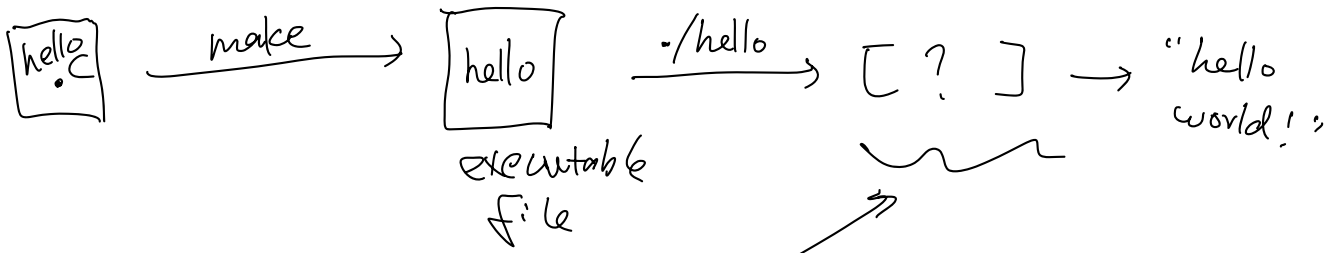
1. main function
 2. intro to processes
 3. Process's view of memory (and registers)
 4. process birth
 5. Shell crash course
 6. Shell internals, part I
 7. File descriptors
 8. Shell internals, part II
-

Recap: C,



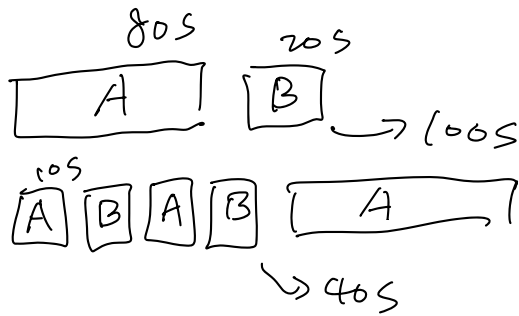
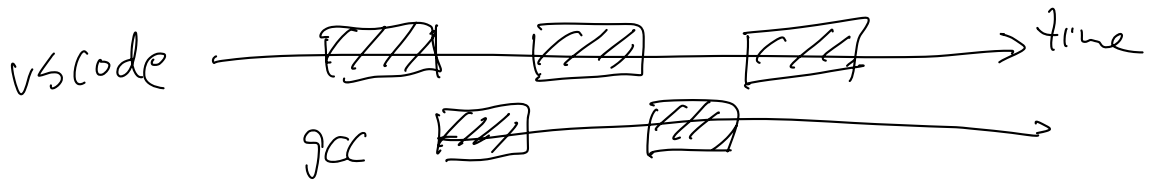
main



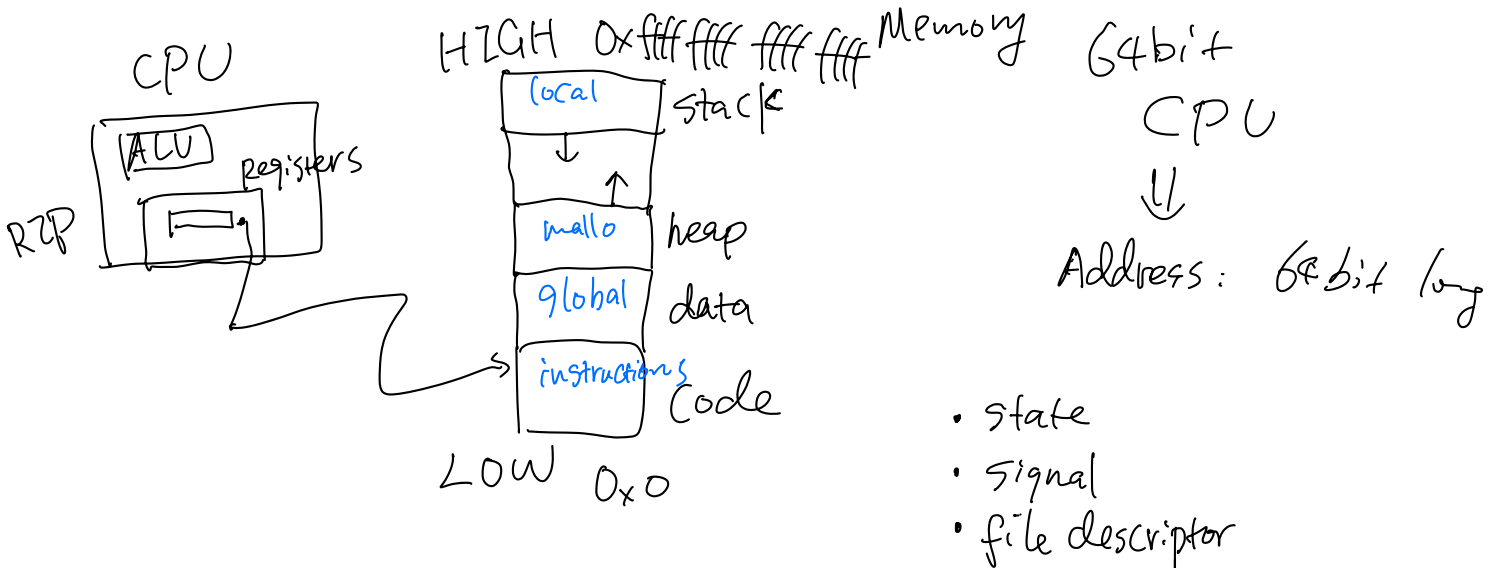


• intro to processes

- ① do multiple things at once
- ② resource efficiency

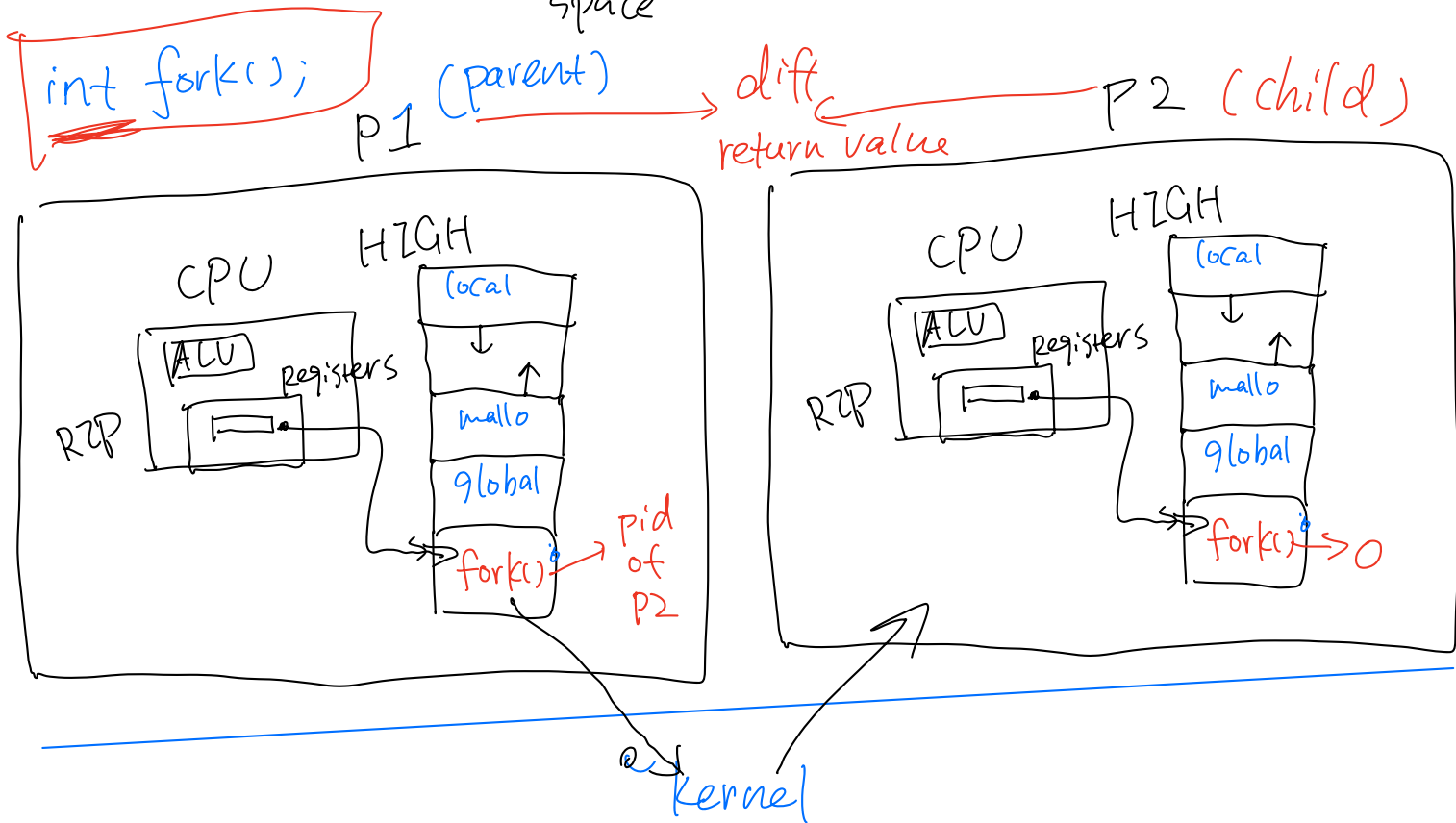
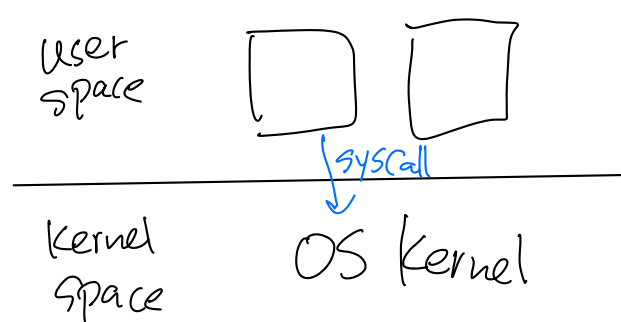


- process = running program.
= an abstract machine.



- state
- signal
- file descriptor

- process birth
- syscall



```

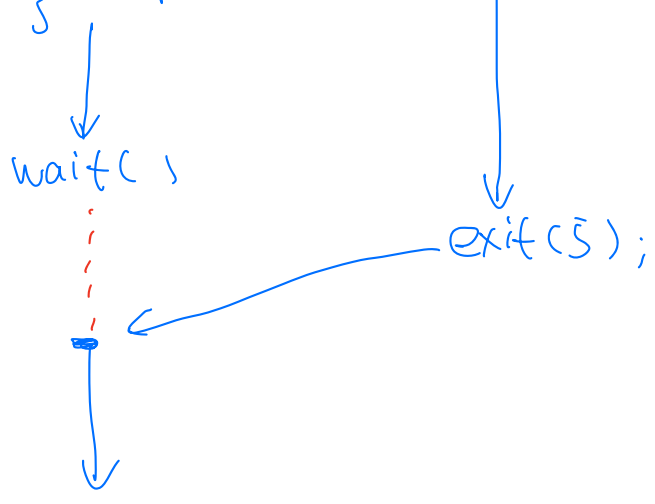
if (fork() == 0) { // who do x? P1 or P2
    do X;
} else { // who do Y? P1, P2, P2
    do Y;
}

```

```

main()
  ↓
fork()
  ↓
P1 ↓ if (ret == 0) {
      printf("I'm child");
} else {
      printf("I'm parent");
}
  ↓
P2 ↓ if (ret == 0) {
      printf("I'm child");
}

```



```
pid_t wait (int *status)
```

- ① Parent block
- ② parent collect exit status of the child

