

			Program Compilation	Undefined Behaviour	Comparing C to Machine Code
Coding Examples					
Jan Faigl					
Department of Computer Science Faculty of Electrical Engineering Czech Technical University in Prague					
Lecture 09					
B3B36PRG – Programming in C					
Jan Faigl, 2024	B3B36PRG – Lecture 09: Coding Examples	1 / 29	Jan Faigl, 2024	B3B36PRG – Lecture 09: Coding Examples	2 / 29
Program Compilation	Undefined Behaviour	Comparing C to Machine Code	Program Compilation	Undefined Behaviour	Comparing C to Machine Code
Arguments of the main() Function			Example of Compilation and Program Execution		
<ul style="list-style-type: none"> During the program execution, the OS passes to the program the number of arguments (<code>argc</code>) and the arguments (<code>argv</code>). <p style="text-align: center;"><i>In the case we are using OS.</i></p>			<ul style="list-style-type: none"> Building the program by the <code>clang</code> compiler – it automatically joins the compilation and linking of the program to the file <code>a.out</code>. <pre>clang var.c</pre>		
<ul style="list-style-type: none"> The first argument is the name of the program. <pre>1 int main(int argc, char *argv[]) 2 { 3 int v; 4 v = 10; 5 v = v + 1; 6 return argc; 7 }</pre> <p style="text-align: right;">lec09/var.c</p>			<ul style="list-style-type: none"> The output file can be specified, e.g., program file <code>var</code>. <pre>clang var.c -o var</pre>		
<ul style="list-style-type: none"> The program is terminated by the <code>return</code> in the <code>main()</code> function. The returned value is passed back to the OS and it can be further use, e.g., to control the program execution. <p style="text-align: center;"><i>Reminder</i></p>			<ul style="list-style-type: none"> Then, the program can be executed as follows. <pre>./var</pre>		
			<ul style="list-style-type: none"> The compilation and execution can be joined to a single command. <pre>clang var.c -o var; ./var</pre>		
			<ul style="list-style-type: none"> The execution can be conditioned to successful compilation. <pre>clang var.c -o var && ./var</pre>		
			<i>Programs return value — 0 means OK.</i> <i>Logical operator && depends on the command interpret, e.g., sh, bash, zsh.</i>		
Jan Faigl, 2024	B3B36PRG – Lecture 09: Coding Examples	5 / 29	Jan Faigl, 2024	B3B36PRG – Lecture 09: Coding Examples	6 / 29
Program Compilation	Undefined Behaviour	Comparing C to Machine Code	Program Compilation	Undefined Behaviour	Comparing C to Machine Code
Example – Processing the Source Code by Preprocessor			Example – Compilation of the Source Code to Assembler		
<ul style="list-style-type: none"> Using the <code>-E</code> flag, we can perform only the preprocessor step. <pre>gcc -E var.c</pre> <p style="text-align: center;"><i>Alternatively clang -E var.c</i></p>			<ul style="list-style-type: none"> Using the <code>-S</code> flag, the source code can be compiled to Assembler. <pre>clang -S var.c -o var.s</pre>		
<pre>1 # 1 "var.c" 2 # 1 "<built-in>" 3 # 1 "<command-line>" 4 # 1 "var.c" 5 int main(int argc, char **argv) { 6 int v; 7 v = 10; 8 v = v + 1; 9 return argc; 10 }</pre> <p style="text-align: right;">lec09/var.c</p>			<pre>1 .file "var.c" 2 .text 3 .globl main 4 .align 16, 0x90 5 .type main,efunction 6 main: # @main 7 .cfi_startproc 8 # BB#0: 9 pushq %rbp 10 .Ltmp2: 11 .cfi_def_cfa_offset 16 12 .Ltmp3: 13 .cfi_offset %rbp, -16 14 movq %rsp, %rbp 15 .Ltmp4: 16 .cfi_def_cfa_register %rbp 17 movl \$0, -4(%rbp) 18 movl %edi, -8(%rbp)</pre>		
			<pre>19 movq %rsi, -16(%rbp) 20 movl \$10, -20(%rbp) 21 movl -20(%rbp), %edi 22 addl \$1, %edi 23 movl %edi, -20(%rbp) 24 movl -8(%rbp), %eax 25 popq %rbp 26 ret 27 .Ltmp5: 28 .size main, .Ltmp5-main 29 .cfi_endproc 30 .ident "FreeBSD clang version 3 31 .4.1 (tags/RELEASE_34/dot1-final 32 208032) 20140512" 33 .section ".note.GNU-stack","",@progbits</pre>		
Jan Faigl, 2024	B3B36PRG – Lecture 09: Coding Examples	8 / 29	Jan Faigl, 2024	B3B36PRG – Lecture 09: Coding Examples	9 / 29
Program Compilation	Undefined Behaviour	Comparing C to Machine Code	Program Compilation	Undefined Behaviour	Comparing C to Machine Code
Part I			Part 1 – Undefined behaviour and inspecting implementation		

Program Compilation Undefined Behaviour Comparing C to Machine Code

Example of Undefined Behaviour

- C standard does not define the behaviour for the overflow of the integer value (`signed`)
 - E.g., for the complement representation, the expression can be
`127 + 1` of the `char` equal to `-128` (see `lec09/demo-loop_byte.c`).
 - Representation of integer values may depend on the architecture and can be different, e.g., when binary or inverse code is used.
- Implementation of the defined behaviour can be computationally expensive, and thus the behaviour is not defined by the standard.
- Behaviour is not defined and depends on the compiler, e.g. `clang` and `gcc` without/with the optimization `-O2`.
 - ```
for (int i = 2147483640; i >= 0; ++i) {
 printf("%i %x\n", i, i);
}
```

`lec09/int_overflow-1.c`
  - Without the optimization, the program prints 8 lines, for `-O2`, the program compiled by `clang` prints 9 lines and `gcc` produces infinite loop.
  - ```
for (int i = 2147483640; i >= 0; i += 4) {
    printf("%i %x\n", i, i);
}
```

`lec09/int_overflow-2.c`
 - Program compiled by `gcc` and `-O2` crashed. *Take a look to the asm code using the compiler parameter -S.*

The screenshot shows the Compiler Explorer interface with the following details:

- Left Panel (Code Editor):** Displays the C code for a square root function. The code includes a main function that calls a square function, which in turn uses a loop to calculate the square root.
- Right Panel (Assembly View):** Shows the generated x86-64 assembly code for the main function. The assembly code includes instructions like push, pop, mov, and div, corresponding to the C code's operations.
- Bottom Panel (Compiler Options):** Provides compiler options for the build, including optimization levels (-O0) and target architectures (x86-64).
- Header Bar:** Includes tabs for "Compiler Explorer", "Undefined Behaviour", and "Comparing C to Machine Code".
- Toolbar:** Features icons for file operations (New, Open, Save, etc.) and other tools.

Program Compilation Undefined Behaviour Comparing C to Machine Code

Compiler Explorer – Analysis of the Optimized Code

- Effect of the code optimization `-O2` on the resulting code that contains undefined behavior (integer overflow).

Code:

```
1 int main(void)
2 {
3     int ret = 8;
4     for (int i = 2147483640; i >= 0; ++i) {
5         ret += i;
6     }
7     return ret;
8 }
```

Compiler Options: x86-64 gcc 12.2 (Ubuntu 12)

Assembly Output:

```
.file "main.c"
1 main:
2     .L1:
3     .L2:
4     .L3:
5     .L4:
6     .L5:
7     .L6:
8     .L7:
9     .L8:
10    .L9:
11    .L10:
12    .L11:
13    .L12:
14    .L13:
15    .L14:
16    .L15:
17    .L16:
18    .L17:
19    .L18:
20    .L19:
21    .L20:
22    .L21:
23    .L22:
24    .L23:
25    .L24:
26    .L25:
27    .L26:
28    .L27:
29    .L28:
30    .L29:
31    .L30:
32    .L31:
33    .L32:
34    .L33:
35    .L34:
36    .L35:
37    .L36:
38    .L37:
39    .L38:
40    .L39:
41    .L40:
42    .L41:
43    .L42:
44    .L43:
45    .L44:
46    .L45:
47    .L46:
48    .L47:
49    .L48:
50    .L49:
51    .L50:
52    .L51:
53    .L52:
54    .L53:
55    .L54:
56    .L55:
57    .L56:
58    .L57:
59    .L58:
60    .L59:
61    .L60:
62    .L61:
63    .L62:
64    .L63:
65    .L64:
66    .L65:
67    .L66:
68    .L67:
69    .L68:
70    .L69:
71    .L70:
72    .L71:
73    .L72:
74    .L73:
75    .L74:
76    .L75:
77    .L76:
78    .L77:
79    .L78:
80    .L79:
81    .L80:
82    .L81:
83    .L82:
84    .L83:
85    .L84:
86    .L85:
87    .L86:
88    .L87:
89    .L88:
90    .L89:
91    .L90:
92    .L91:
93    .L92:
94    .L93:
95    .L94:
96    .L95:
97    .L96:
98    .L97:
99    .L98:
100   .L99:
101   .L100:
102   .L101:
103   .L102:
104   .L103:
105   .L104:
106   .L105:
107   .L106:
108   .L107:
109   .L108:
110   .L109:
111   .L110:
112   .L111:
113   .L112:
114   .L113:
115   .L114:
116   .L115:
117   .L116:
118   .L117:
119   .L118:
120   .L119:
121   .L120:
122   .L121:
123   .L122:
124   .L123:
125   .L124:
126   .L125:
127   .L126:
128   .L127:
129   .L128:
130   .L129:
131   .L130:
132   .L131:
133   .L132:
134   .L133:
135   .L134:
136   .L135:
137   .L136:
138   .L137:
139   .L138:
140   .L139:
141   .L140:
142   .L141:
143   .L142:
144   .L143:
145   .L144:
146   .L145:
147   .L146:
148   .L147:
149   .L148:
150   .L149:
151   .L150:
152   .L151:
153   .L152:
154   .L153:
155   .L154:
156   .L155:
157   .L156:
158   .L157:
159   .L158:
160   .L159:
161   .L160:
162   .L161:
163   .L162:
164   .L163:
165   .L164:
166   .L165:
167   .L166:
168   .L167:
169   .L168:
170   .L169:
171   .L170:
172   .L171:
173   .L172:
174   .L173:
175   .L174:
176   .L175:
177   .L176:
178   .L177:
179   .L178:
180   .L179:
181   .L180:
182   .L181:
183   .L182:
184   .L183:
185   .L184:
186   .L185:
187   .L186:
188   .L187:
189   .L188:
190   .L189:
191   .L190:
192   .L191:
193   .L192:
194   .L193:
195   .L194:
196   .L195:
197   .L196:
198   .L197:
199   .L198:
200   .L199:
201   .L200:
202   .L201:
203   .L202:
204   .L203:
205   .L204:
206   .L205:
207   .L206:
208   .L207:
209   .L208:
210   .L209:
211   .L210:
212   .L211:
213   .L212:
214   .L213:
215   .L214:
216   .L215:
217   .L216:
218   .L217:
219   .L218:
220   .L219:
221   .L220:
222   .L221:
223   .L222:
224   .L223:
225   .L224:
226   .L225:
227   .L226:
228   .L227:
229   .L228:
230   .L229:
231   .L230:
232   .L231:
233   .L232:
234   .L233:
235   .L234:
236   .L235:
237   .L236:
238   .L237:
239   .L238:
240   .L239:
241   .L240:
242   .L241:
243   .L242:
244   .L243:
245   .L244:
246   .L245:
247   .L246:
248   .L247:
249   .L248:
250   .L249:
251   .L250:
252   .L251:
253   .L252:
254   .L253:
255   .L254:
256   .L255:
257   .L256:
258   .L257:
259   .L258:
260   .L259:
261   .L260:
262   .L261:
263   .L262:
264   .L263:
265   .L264:
266   .L265:
267   .L266:
268   .L267:
269   .L268:
270   .L269:
271   .L270:
272   .L271:
273   .L272:
274   .L273:
275   .L274:
276   .L275:
277   .L276:
278   .L277:
279   .L278:
280   .L279:
281   .L280:
282   .L281:
283   .L282:
284   .L283:
285   .L284:
286   .L285:
287   .L286:
288   .L287:
289   .L288:
290   .L289:
291   .L290:
292   .L291:
293   .L292:
294   .L293:
295   .L294:
296   .L295:
297   .L296:
298   .L297:
299   .L298:
300   .L299:
301   .L300:
302   .L301:
303   .L302:
304   .L303:
305   .L304:
306   .L305:
307   .L306:
308   .L307:
309   .L308:
310   .L309:
311   .L310:
312   .L311:
313   .L312:
314   .L313:
315   .L314:
316   .L315:
317   .L316:
318   .L317:
319   .L318:
320   .L319:
321   .L320:
322   .L321:
323   .L322:
324   .L323:
325   .L324:
326   .L325:
327   .L326:
328   .L327:
329   .L328:
330   .L329:
331   .L330:
332   .L331:
333   .L332:
334   .L333:
335   .L334:
336   .L335:
337   .L336:
338   .L337:
339   .L338:
340   .L339:
341   .L340:
342   .L341:
343   .L342:
344   .L343:
345   .L344:
346   .L345:
347   .L346:
348   .L347:
349   .L348:
350   .L349:
351   .L350:
352   .L351:
353   .L352:
354   .L353:
355   .L354:
356   .L355:
357   .L356:
358   .L357:
359   .L358:
360   .L359:
361   .L360:
362   .L361:
363   .L362:
364   .L363:
365   .L364:
366   .L365:
367   .L366:
368   .L367:
369   .L368:
370   .L369:
371   .L370:
372   .L371:
373   .L372:
374   .L373:
375   .L374:
376   .L375:
377   .L376:
378   .L377:
379   .L378:
380   .L379:
381   .L380:
382   .L381:
383   .L382:
384   .L383:
385   .L384:
386   .L385:
387   .L386:
388   .L387:
389   .L388:
390   .L389:
391   .L390:
392   .L391:
393   .L392:
394   .L393:
395   .L394:
396   .L395:
397   .L396:
398   .L397:
399   .L398:
400   .L399:
401   .L400:
402   .L401:
403   .L402:
404   .L403:
405   .L404:
406   .L405:
407   .L406:
408   .L407:
409   .L408:
410   .L409:
411   .L410:
412   .L411:
413   .L412:
414   .L413:
415   .L414:
416   .L415:
417   .L416:
418   .L417:
419   .L418:
420   .L419:
421   .L420:
422   .L421:
423   .L422:
424   .L423:
425   .L424:
426   .L425:
427   .L426:
428   .L427:
429   .L428:
430   .L429:
431   .L430:
432   .L431:
433   .L432:
434   .L433:
435   .L434:
436   .L435:
437   .L436:
438   .L437:
439   .L438:
440   .L439:
441   .L440:
442   .L441:
443   .L442:
444   .L443:
445   .L444:
446   .L445:
447   .L446:
448   .L447:
449   .L448:
450   .L449:
451   .L450:
452   .L451:
453   .L452:
454   .L453:
455   .L454:
456   .L455:
457   .L456:
458   .L457:
459   .L458:
460   .L459:
461   .L460:
462   .L461:
463   .L462:
464   .L463:
465   .L464:
466   .L465:
467   .L466:
468   .L467:
469   .L468:
470   .L469:
471   .L470:
472   .L471:
473   .L472:
474   .L473:
475   .L474:
476   .L475:
477   .L476:
478   .L477:
479   .L478:
480   .L479:
481   .L480:
482   .L481:
483   .L482:
484   .L483:
485   .L484:
486   .L485:
487   .L486:
488   .L487:
489   .L488:
490   .L489:
491   .L490:
492   .L491:
493   .L492:
494   .L493:
495   .L494:
496   .L495:
497   .L496:
498   .L497:
499   .L498:
500   .L499:
501   .L500:
502   .L501:
503   .L502:
504   .L503:
505   .L504:
506   .L505:
507   .L506:
508   .L507:
509   .L508:
510   .L509:
511   .L510:
512   .L511:
513   .L512:
514   .L513:
515   .L514:
516   .L515:
517   .L516:
518   .L517:
519   .L518:
520   .L519:
521   .L520:
522   .L521:
523   .L522:
524   .L523:
525   .L524:
526   .L525:
527   .L526:
528   .L527:
529   .L528:
530   .L529:
531   .L530:
532   .L531:
533   .L532:
534   .L533:
535   .L534:
536   .L535:
537   .L536:
538   .L537:
539   .L538:
540   .L539:
541   .L540:
542   .L541:
543   .L542:
544   .L543:
545   .L544:
546   .L545:
547   .L546:
548   .L547:
549   .L548:
550   .L549:
551   .L550:
552   .L551:
553   .L552:
554   .L553:
555   .L554:
556   .L555:
557   .L556:
558   .L557:
559   .L558:
560   .L559:
561   .L560:
562   .L561:
563   .L562:
564   .L563:
565   .L564:
566   .L565:
567   .L566:
568   .L567:
569   .L568:
570   .L569:
571   .L570:
572   .L571:
573   .L572:
574   .L573:
575   .L574:
576   .L575:
577   .L576:
578   .L577:
579   .L578:
580   .L579:
581   .L580:
582   .L581:
583   .L582:
584   .L583:
585   .L584:
586   .L585:
587   .L586:
588   .L587:
589   .L588:
590   .L589:
591   .L590:
592   .L591:
593   .L592:
594   .L593:
595   .L594:
596   .L595:
597   .L596:
598   .L597:
599   .L598:
5100  .L599:
5101  .L600:
5102  .L601:
5103  .L602:
5104  .L603:
5105  .L604:
5106  .L605:
5107  .L606:
5108  .L607:
5109  .L608:
5110  .L609:
5111  .L610:
5112  .L611:
5113  .L612:
5114  .L613:
5115  .L614:
5116  .L615:
5117  .L616:
5118  .L617:
5119  .L618:
5120  .L619:
5121  .L620:
5122  .L621:
5123  .L622:
5124  .L623:
5125  .L624:
5126  .L625:
5127  .L626:
5128  .L627:
5129  .L628:
5130  .L629:
5131  .L630:
5132  .L631:
5133  .L632:
5134  .L633:
5135  .L634:
5136  .L635:
5137  .L636:
5138  .L637:
5139  .L638:
5140  .L639:
5141  .L640:
5142  .L641:
5143  .L642:
5144  .L643:
5145  .L644:
5146  .L645:
5147  .L646:
5148  .L647:
5149  .L648:
5150  .L649:
5151  .L650:
5152  .L651:
5153  .L652:
5154  .L653:
5155  .L654:
5156  .L655:
5157  .L656:
5158  .L657:
5159  .L658:
5160  .L659:
5161  .L660:
5162  .L661:
5163  .L662:
5164  .L663:
5165  .L664:
5166  .L665:
5167  .L666:
5168  .L667:
5169  .L668:
5170  .L669:
5171  .L670:
5172  .L671:
5173  .L672:
5174  .L673:
5175  .L674:
5176  .L675:
5177  .L676:
5178  .L677:
5179  .L678:
5180  .L679:
5181  .L680:
5182  .L681:
5183  .L682:
5184  .L683:
5185  .L684:
5186  .L685:
5187  .L686:
5188  .L687:
5189  .L688:
5190  .L689:
5191  .L690:
5192  .L691:
5193  .L692:
5194  .L693:
5195  .L694:
5196  .L695:
5197  .L696:
5198  .L697:
5199  .L698:
5200  .L699:
5201  .L700:
5202  .L701:
5203  .L702:
5204  .L703:
5205  .L704:
5206  .L705:
5207  .L706:
5208  .L707:
5209  .L708:
5210  .L709:
5211  .L710:
5212  .L711:
5213  .L712:
5214  .L713:
5215  .L714:
5216  .L715:
5217  .L716:
5218  .L717:
5219  .L718:
5220  .L719:
5221  .L720:
5222  .L721:
5223  .L722:
5224  .L723:
5225  .L724:
5226  .L725:
5227  .L726:
5228  .L727:
5229  .L728:
5230  .L729:
5231  .L730:
5232  .L731:
5233  .L732:
5234  .L733:
5235  .L734:
5236  .L735:
5237  .L736:
5238  .L737:
5239  .L738:
5240  .L739:
5241  .L740:
5242  .L741:
5243  .L742:
5244  .L743:
5245  .L744:
5246  .L745:
5247  .L746:
5248  .L747:
5249  .L748:
5250  .L749:
5251  .L750:
5252  .L751:
5253  .L752:
5254  .L753:
5255  .L754:
5256  .L755:
5257  .L756:
5258  .L757:
5259  .L758:
5260  .L759:
5261  .L760:
5262  .L761:
5263  .L762:
5264  .L763:
5265  .L764:
5266  .L765:
5267  .L766:
5268  .L767:
5269  .L768:
5270  .L769:
5271  .L770:
5272  .L771:
5273  .L772:
5274  .L773:
5275  .L774:
5276  .L775:
5277  .L776:
5278  .L777:
5279  .L778:
5280  .L779:
5281  .L780:
5282  .L781:
5283  .L782:
5284  .L783:
5285  .L784:
5286  .L785:
5287  .L786:
5288  .L787:
5289  .L788:
5290  .L789:
5291  .L790:
5292  .L791:
5293  .L792:
5294  .L793:
5295  .L794:
5296  .L795:
5297  .L796:
5298  .L797:
5299  .L798:
5300  .L799:
5301  .L800:
5302  .L801:
5303  .L802:
5304  .L803:
5305  .L804:
5306  .L805:
5307  .L806:
5308  .L807:
5309  .L808:
5310  .L809:
5311  .L810:
5312  .L811:
5313  .L812:
5314  .L813:
5315  .L814:
5316  .L815:
5317  .L816:
5318  .L817:
5319  .L818:
5320  .L819:
5321  .L820:
5322  .L821:
5323  .L822:
5324  .L823:
5325  .L824:
5326  .L825:
5327  .L826:
5328  .L827:
5329  .L828:
5330  .L829:
5331  .L830:
5332  .L831:
5333  .L832:
5334  .L833:
5335  .L834:
5336  .L835:
5337  .L836:
5338  .L837:
5339  .L838:
5340  .L839:
5341  .L840:
5342  .L841:
5343  .L842:
5344  .L843:
5345  .L844:
5346  .L845:
5347  .L846:
5348  .L847:
5349  .L848:
5350  .L849:
5351  .L850:
5352  .L851:
5353  .L852:
5354  .L853:
5355  .L854:
5356  .L855:
5357  .L856:
5358  .L857:
5359  .L858:
5360  .L859:
5361  .L860:
5362  .L861:
5363  .L862:
5364  .L863:
5365  .L864:
5366  .L865:
5367  .L866:
5368  .L867:
5369  .L868:
5370  .L869:
5371  .L870:
5372  .L871:
5373  .L872:
5374  .L873:
5375  .L874:
5376  .L875:
5377  .L876:
5378  .L877:
5379  .L878:
5380  .L879:
5381  .L880:
5382  .L881:
5383  .L882:
5384  .L883:
5385  .L884:
5386  .L885:
5387  .L886:
5388  .L887:
5389  .L888:
5390  .L889:
5391  .L890:
5392  .L891:
5393  .L892:
5394  .L893:
5395  .L894:
5396  .L895:
5397  .L896:
5398  .L897:
5399  .L898:
5400  .L899:
5401  .L900:
5402  .L901:
5403  .L902:
5404  .L903:
5405  .L904:
5406  .L905:
5407  .L906:
5408  .L907:
5409  .L908:
5410  .L909:
5411  .L910:
5412  .L911:
5413  .L912:
5414  .L913:
5415  .L914:
5416  .L915:
5417  .L916:
5418  .L917:
5419  .L918:
5420  .L919:
5421  .L920:
5422  .L921:
5423  .L922:
5424  .L923:
5425  .L924:
5426  .L925:
5427  .L926:
5428  .L927:
5429  .L928:
5430  .L929:
5431  .L930:
5432  .L931:
5433  .L932:
5434  .L933:
5435  .L934:
5436  .L935:
5437  .L936:
5438  .L937:
5439  .L938:
5440  .L939:
5441  .L940:
5442  .L941:
5443  .L942:
5444  .L943:
5445  .L944:
5446  .L945:
5447  .L946:
5448  .L947:
5449  .L948:
5450  .L949:
5451  .L950:
5452  .L951:
5453  .L952:
5454  .L953:
5455  .L954:
5456  .L955:
5457  .L956:
5458  .L957:
5459  .L958:
5460  .L959:
5461  .L960:
5462  .L961:
5463  .L962:
5464  .L963:
5465  .L964:
5466  .L965:
5467  .L966:
5468  .L967:
5469  .L968:
5470  .L969:
5471  .L970:
5472  .L971:
5473  .L972:
5474  .L973:
5475  .L974:
5476  .L975:
5477  .L976:
5478  .L977:
5479  .L978:
5480  .L979:
5481  .L980:
5482  .L981:
5483  .L982:
5484  .L983:
5485  .L984:
5486  .L985:
5487  .L986:
5488  .L987:
5489  .L988:
5490  .L989:
5491  .L990:
5492  .L991:
5493  .L992:
5494  .L993:
5495  .L994:
5496  .L995:
5497  .L996:
5498  .L997:
5499  .L998:
5500  .L999:
5501  .L1000:
5502  .L1001:
5503  .L1002:
5504  .L1003:
5505  .L1004:
5506  .L1005:
5507  .L1006:
5508  .L1007:
5509  .L1008:
5510  .L1009:
5511  .L1010:
5512  .L1011:
5513  .L1012:
5514  .L1013:
5515  .L1014:
5516  .L1015:
5517  .L1016:
5518  .L1017:
5519  .L1018:
5520  .L1019:
5521  .L1020:
5522  .L1021:
5523  .L1022:
5524  .L1023:
5525  .L1024:
5526  .L1025:
5527  .L1026:
5528  .L1027:
5529  .L1028:
5530  .L1029:
5531  .L1030:
5532  .L1031:
5533  .L1032:
5534  .L1033:
5535  .L1034:
5536  .L1035:
5537  .L1036:
5538  .L1037:
5539  .L1038:
5540  .L1039:
5541  .L1040:
5542  .L1041:
5543  .L1042:
5544  .L1043:
5545  .L1044:
5546  .L1045:
5547  .L1046:
5548  .L1047:
5549  .L1048:
5550  .L1049:
5551  .L1050:
5552  .L1051:
5553  .L1052:
5554  .L1053:
5555  .L1054:
5556  .L1055:
5557  .L1056:
5558  .L1057:
5559  .L1058:
5560  .L1059:
5561  .L1060:
5562  .L1061:
5563  .L1062:
5564  .L1063:
5565  .L1064:
5566  .L1065:
5567  .L1066:
5568  .L1067:
5569  .L1068:
5570  .L1069:
5571  .L1070:
5572  .L1071:
5573  .L1072:
5574  .L1073:
5575  .L1074:
5576  .L1075:
5577  .L1076:
5578  .L1077:
5579  .L1078:
5580  .L1079:
5581  .L1080:
5582  .L1081:
5583  .L1082:
5584  .L1083:
5585  .L1084:
5586  .L1085:
5587  .L1086:
5588  .L1087:
5589  .L1088:
5590  .L1089:
5591  .L1090:
5592  .L1091:
5593  .L1092:
5594  .L1093:
5595  .L1094:
5596  .L1095:
5597  .L1096:
5598  .L1097:
5599  .L1098:
5600  .L1099:
5601  .L1100:
5602  .L1101:
5603  .L1102:
5604  .L1103:
5605  .L1104:
5606  .L1105:
5607  .L1106:
5608  .L1107:
5609  .L1108:
5610  .L1109:
5611  .L1110:
5612  .L1111:
5613  .L1112:
5614  .L1113:
5615  .L1114:
5616  .L1115:
5617  .L1116:
5618  .L1117:
5619  .L1118:
5620  .L1119:
5621  .L1120:
5622  .L1121:
5623  .L1122:
5624  .L1123:
5625  .L1124:
5626  .L1125:
5627  .L1126:
5628  .L1127:
5629  .L1128:
5630  .L1129:
5631  .L1130:
5632  .L1131:
5633  .L1132:
5634  .L1133:
5635  .L1134:
5636  .L1135:
5637  .L1136:
5638  .L1137:
5639  .L1138:
5640  .L1139:
5641  .L1140:
5642  .L1141:
5643  .L1142:
5644  .L1143:
5645  .L1144:
5646  .L1145:
5647  .L1146:
5648  .L1147:
5649  .L1148:
5650  .L1149:
5651  .L1150:
5652  .L1151:
5653  .L1152:
5654  .L1153:
5655  .L1154:
5656  .L1155:
5657  .L1156:
5658  .L1157:
5
```

Jan Faigl, 2024 B3B36PRG – Lecture 09: Coding Examples 13 /

Jan Faigl, 2024 B3B36PRG – Lecture 09: Coding Examples 14 / 24

Debugging

Debugging the Code

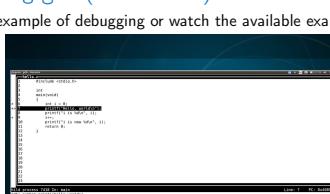
- Principally there are two ways of debugging: **stepping** (program animation) and **logging**.
- **Stepping** is interactive debugging that might be suitable for relatively small, less complex codes, and non real-time applications.
 - In stepping, we use **breakpoints**, **watches** to stop the program execution at certain conditions and then inspect variables and stepping next instructions.
 - In C, most of the visual interfaces uses **gdb**.
 - It might be suitable to compile the program with **debugging information**, e.g., using **-g flag**.
`clang -g main.c -o main`
- **Logging** can range from simple print messages to **stderr** to sophisticated **loggers**, such as **log4c**.
- We can further enjoy tools such as **valgrind** for dynamic analysis, specifically for bugs in memory access.
For more than 20 years, see <https://valgrind.org/>.

Jan Fagl, 2024 B3B36PRG – Lecture 09: Coding Examples 16 / 25

Debugging

Debugging using gdb (or VS Code)

- Interactive example of debugging or watch the available examples and tutorials.



cppcon@



GREG LAW

Give me fifteen minutes and I'll change your view of GDB

www.CppCon.org

- CppCon 2015: Greg Law "Give me 15 minutes & I'll change your view of GDB."

<https://www.youtube.com/watch?v=PorfLSr3DDI>

Jan Faigl, 2024 B3B36PRG – Lecture 09: Coding Examples 17

Debugging

Example of using valgrind

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(void)
4 {
5     int *a = malloc(2 * sizeof(*a));
6     for (int i = 0; i < 3; ++i) {
7         a[i] = i;
8     }
9     for (int i = 0; i < 3; ++i) {
10        printf("%d\n", a[i]);
11    }
12    //free(a);
13    return 0;
14 }
15 }
```

```
$ clang -g mem_val.c -o mem_val
$ valgrind ./mem_val
...
==87826== Invalid write of size 4
==87826== at 0x201999: main (mem_val.c:9)
==87826== Address 0x5400048 is 0 bytes after
          a block of size 8 alloc'd
==87826== at 0x4853874: malloc (in /usr/
          local/libexec/valgrind/vgpreload_memcheck-
          amd64-freebsd.so)
==87826== by 0x201978: main (mem_val.c:6)
==87826== by 0x201978: main (mem_val.c:6)
...
0
```

le09/mem_val.c

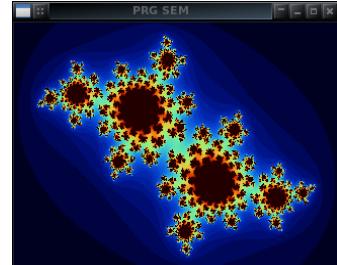
Jan Faigl, 2024 B3B36PRG – Lecture 09: Coding Examples 19 / 25

Debugging

Example of malloc failure

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(void)
4 {
5     const size_t size = 20 * 1024 * 1024; // 20 MB
6     size_t *a = malloc(size * sizeof(*a)); // 20 MB * sizeof(long)
7     if (!a) {
8         fprintf(stderr, "ERROR: malloc failed\n");
9     }
10    return -1;
11 }
12 for (size_t i = 0; i < size; ++i) {
13     a[i] = i;
14 }
15 fprintf(stderr, "INFO: array of %lu\n"
16         "%lu values initialized.\n", size);
17 free(a);
18 }
19 }
```

\$ clang mem_fail.c -o mem_fail
\$ bash
\$ ulimit -d 10 -m 10 -v 1000000 -w 0
\$./mem_fail
INFO: array of 20971520 size_t values initialized.
\$ exit
exit
\$ bash
\$ ulimit -d 10 -m 10 -v 10000 -w 0
\$./mem_fail
ERROR: malloc failed!

<p>Named pipes</p> <h2>Part III</h2> <h3>Part 3 – Examples</h3>	<p>Multithread Applications – Semestral Project</p> <h4>Communication using Named Pipes</h4> <ul style="list-style-type: none"> ■ Implement two applications <code>main</code> and <code>module</code> that communicates through named pipes. <code>lec09/pipes/create_pipes.sh</code> <code>lec09/pipes/prg_lec09_main.c</code>, <code>lec09/pipes/prg-lec09-module.c</code> ■ <code>module</code> opens pipe <code>/tmp/prg-lec09.pipe</code> for reading. ■ <code>main</code> opens pipe <code>/tmp/prg-lec09.pipe</code> for writing. ■ The applications communicate using simple character-oriented protocol. <ul style="list-style-type: none"> ■ 's' – stop. ■ 'e' – enable (start). ■ 'b' – bye. ■ '1'–'5' – set sleep period to 50 ms, 100 ms, 200 ms, 500 ms, 1000 ms. ■ The pipe can be opened using functions from the <code>prg.io_nonblock</code> library. <code>lec09/pipes/prg_io_nonblock.h</code>, <code>lec09/pipes/prg-io_nonblock.c</code> ■ Examine the provided code and test it. <p><i>The example is without threads.</i></p>	<p>Named pipes</p> <p>Multithread Applications – Semestral Project</p> <h4>Remote Control of Computational Application (Module) – Semestral Project</h4> <ul style="list-style-type: none"> ■ Implement multi-thread application with separate threads for sources of asynchronous events. <ul style="list-style-type: none"> ■ User input from <code>stdin</code> (<code>keyboard</code>). ■ Pipe reading from the computational module. ■ Use simple visualization using <code>sdl</code>. ■ Implement the main program logic in the main (<code>boss</code>) thread using <code>event queue</code>. <ul style="list-style-type: none"> ■ The main thread reads from the queue. ■ The secondary threads (<code>keyboard</code> and <code>pipe</code>) write to the queue. ■ The main thread manages output resources (<code>visualization</code>, <code>write to pipe</code>). <ul style="list-style-type: none"> ■ Eventually also <code>stdout</code> or even <code>stderr</code>, which is, however, not required. ■ Use the example of multi-thread application from Lecture 8. https://cv.fel.cvut.cz/wiki/courses/b3b36prg/sementral-project/start
<p>Jan Faigl, 2024</p> <p>B3B36PRG – Lecture 09: Coding Examples</p> <p>23 / 29</p> <p>Topics Discussed</p>	<p>Jan Faigl, 2024</p> <p>B3B36PRG – Lecture 09: Coding Examples</p> <p>23 / 29</p> <p>Topics Discussed</p> <p>Summary of the Lecture</p> <ul style="list-style-type: none"> ■ Program compilation. ■ Undefined behaviour. ■ Comments on debugging. ■ Named pipes. ■ Semestral project. 	<p>Jan Faigl, 2024</p> <p>B3B36PRG – Lecture 09: Coding Examples</p> <p>25 / 29</p> <p>Jan Faigl, 2024</p> <p>B3B36PRG – Lecture 09: Coding Examples</p> <p>27 / 29</p> 
<p>Jan Faigl, 2024</p> <p>B3B36PRG – Lecture 09: Coding Examples</p> <p>28 / 29</p>	<p>Jan Faigl, 2024</p> <p>B3B36PRG – Lecture 09: Coding Examples</p> <p>28 / 29</p>	<p>Jan Faigl, 2024</p> <p>B3B36PRG – Lecture 09: Coding Examples</p> <p>29 / 29</p>