

```

        extract_num-
ber_and_incr (destination, source) int
    *destination; unsigned char **source; { extract_num-
ber (destination, *source); *source += 2; } #ifndef EXTRACT_MAC-
ROS #undef EXTRACT_NUMBER_AND_INCR #define EXTRACT_NUM-
BER_AND_INCR(dest, src) \extract_number_and_incr (&dest, &src) #endif /*
not EXTRACT_MACROS */ #endif /* DEBUG */ /* If DEBUG is defined, Regexp prints
many voluminous messages about what it is doing (if the variable 'debug' is nonzero). If
linked with the main program in 'iregex.c', you can enter patterns and strings interactively.
And if linked with the main program in 'main.c' and the other test files, you can run the al-
ready-written tests. */ #ifdef DEBUG /* We use standard I/O for debugging. */ #include <stdio.h>
/* It is useful to test things that "must" be true when debugging. */ #include <assert.h> static int
debug = 0; #define DEBUG_STATEMENT(e) #define DEBUG_PRINT1(x) if (debug) printf (x) #define
DEBUG_PRINT2(x1, x2) if (debug) printf (x1, x2) #define DEBUG_PRINT3(x1, x2, x3) if (debug) printf
(x1, x2, x3) #define DEBUG_PRINT4(x1, x2, x3, x4) if (debug) printf (x1, x2, x3, x4) #define DE-
BUG_PRINT_COMPILED_PATTERN(p, s, e) if (debug) print_partial_compiled_pattern (s, e) #define DE-
BUG_PRINT_DOUBLE_STRING(w, s1, sz1, s2, sz2) \ if (debug) print_double_string (w, s1, sz1, s2, sz2)
extern void printchar(); /* Print the fastmap in human-readable form. */ void print_fastmap (fastmap)
char *fastmap; { unsigned was_a_range = 0; unsigned i = 0; while (i < (1 << BYTEWIDTH)) { if (fastmap[i++]
{ was_a_range = 0; printchar (i - 1); while (i < (1 << BYTEWIDTH) && fastmap[i]) { was_a_range = 1; i++; } if
(was_a_range) { printf ("-"); printchar (i - 1); } } putchar ('\n'); } /* Print a compiled pattern string in hu-
man-readable form, starting at the START pointer into it and ending just before the pointer END. */ void
print_partial_compiled_pattern (start, end) unsigned char *start; unsigned char *end; { int mcnt, mcnt2; un-
signed char *p = start; unsigned char *pend = end; if (start == NULL) { printf ("(null)\n"); return; } /* Loop over
pattern commands. */ while (p < pend) { switch ((re_opcode_t) *p++) { case no_op: printf ("/no_op");
break; case exactn: mcnt = *p++; printf ("/exactn/%d", mcnt); do { putchar ('/'); printchar (*p++); }
while (--mcnt); break; case start_memory: mcnt = *p++; printf ("/start_memory/%d/%d", mcnt,
*p++); break; case stop_memory: mcnt = *p++; printf ("/stop_memory/%d/%d", mcnt, *p++);
break; case duplicate: printf ("/duplicate/%d", *p++); break; case anychar: printf ("/anychar");
break; case charset: case charset_not: { register int c; printf ("/charset%s", (re_opcode_t) *(p -
1) == charset_not ? "_not" : ""); assert (p + *p < pend); for (c = 0; c < *p; c++) { unsigned bit;
unsigned char map_byte = p[1 + c]; putchar ('/'); for (bit = 0; bit < BYTEWIDTH; bit++) if
(map_byte & (1 << bit)) printchar (c * BYTEWIDTH + bit); } p += 1 + *p; break; } case beg-
line: printf ("/begline"); break; case newline: printf ("/newline"); break; case on_failure_-
jump: extract_number_and_incr (&mcnt, &p); printf ("/on_failure_jump/0/%d", mcnt);
break; case on_failure_keep_string_jump: extract_number_and_incr (&mcnt, &p); printf
("/on_failure_keep_string_jump/0/%d", mcnt); break; case dummy_failure_jump: ex-
tract_number_and_incr (&mcnt, &p); printf ("/dummy_failure_jump/0/%d", mcnt); break;
case push_dummy_failure: printf ("/push_dummy_failure"); break; case may-
be_pop_jump: extract_number_and_incr (&mcnt, &p); printf
("/maybe_pop_jump/0/%d", mcnt); break; case pop_failure_-
jump: extract_number_and_incr (&mcnt, &p); printf ("/pop_-
failure_jump/0/%d", mcnt); break; case jump_past_alt:
extract_number_and_incr (&mcnt, &p); printf ("/-

```

C. BR.

EMPFEHLUNGEN FÜR DEN EINSATZ SOZIALER
NETZWERKE IM KATASTROPHENSCHUTZ

KRISEN MIT SOCIAL MEDIA
INTELLIGENCE BEWÄLTIGEN

C. Br.

**Krisen mit Social Media
Intelligence bewältigen**

**Empfehlungen für den Einsatz sozialer
Netzwerke im Katastrophenschutz**

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Impressum:

Copyright © Studylab 2020

Ein Imprint der GRIN Publishing GmbH, München

Druck und Bindung: Books on Demand GmbH, Norderstedt, Germany

Coverbild: GRIN Publishing GmbH | Freepik.com | Flaticon.com | ei8htz

Inhaltsverzeichnis

Abkürzungsverzeichnis	IV
Abbildungsverzeichnis	VI
1 Einleitung	1
2 Forschungsmethode	2
3 Begriffsbestimmungen und Grundlagen	3
3.1 Krisen und Katastrophen.....	3
3.2 Begriff des Informationssystems	7
3.3 Begriff des Input-Output-Systems	8
4 Warnung der Bevölkerung als Input-Output-System	9
4.1 Inputquellen.....	9
4.2 Verarbeitung des Inputs.....	14
4.3 Outputquellen.....	15
5 Analyse des Potentials von Social Media bei der Krisenbewältigung	21
5.1 Nutzungsmuster von Social Media Plattformen	22
5.2 Social Media Intelligence	24
5.3 Partizipation durch freiwillige Helfer	32
6 Fazit	38
7 Limitationen und Ausblick	41
Literaturverzeichnis	43

Abkürzungsverzeichnis

A2A	Authorities to Authorities
A2C	Authorities to Citizens
AML	Advanced Mobile Location
API	Application Programming Interface
BAO	Besondere Aufbauorganisation
BBK	Bundesamt für Bevölkerungsschutz und Katastrophenhilfe
BMI	Bundesministerium des Innern
BSH	Bundesamt für Seeschifffahrt und Hydrografie
BSI	Bundesamt für Sicherheit in der Informationstechnik
C2A	Citizens to Authorities
C2C	Citizens to Citizens
CB	Cell Broadcast
CBRN	Chemisch, biologisch, radiologisch, nuklear
DWD	Deutscher Wetterdienst
EAS	Emergency Alert System
EENA	European Emergency Number Association
EK	Europäische Kommission
FEMA	Federal Emergency Management Agency
GITEWS	Deutsch-Indonesisches Tsunami Frühwarnsystem
GMLZ	Gemeinsames Melde- und Lagezentrum von Bund und Ländern
IoT	Internet of Things
IS	Informationssystem
KFÜ	Kernreaktorfernüberwachung
MCI	Mensch-Computer-Interaktion
MoWaS	Modulares Warnsystem
NINA	Notfall-Informations- und Nachrichten-App